UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Construcción de modelos de clasificación automática para la detección del acoso

Alejandra De Francisco Rus
Tutora: Rosa María Carro Salas

Mayo 2018

Construcción de modelos de clasificación automática para la detección del acoso

AUTOR: Alejandra De Francisco Rus
TUTORA: Rosa María Carro Salas

Grupo de Herramientas Interactivas Avanzadas
Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2018

Resumen

El acoso escolar es un problema cada vez más visible en las aulas de los colegios, y preocupa a expertos de todos los ámbitos. Las graves consecuencias que genera a las víctimas sugieren la necesidad de estudiarlo con el fin de evitar nuevos casos.

Además, el uso de las tecnologías y las redes sociales crece progresivamente en jóvenes y adolescentes. Esta nueva manera de comunicación ha propiciado la transición del acoso escolar únicamente a la práctica conjunta de acoso tradicional y el presente a través de los medios tecnológicos, el *ciberbullying.*

Este Trabajo Fin de Grado tiene como objetivo principal desarrollar un clasificador supervisado que sea capaz de decidir si un texto sería calificado como *bullying* o no por un experto en este ámbito. Para conseguir esta meta, se ha trabajado con un conjunto de documentos obtenidos de Twitter, que han sido marcados por un etiquetador. Después, se ha aplicado un proceso de limpieza y tratado de estos *tuits*, en el que se han eliminado emoticonos, nombres de usuarios y retuits. También se han agrupado conjuntos de términos con el mismo significado en otras expresiones, para hacer un estudio más genérico. Más tarde, se han aplicado procedimientos para convertir estos textos ya tratados a vectores numéricos, de forma que sirvan como argumentos de entrada para los clasificadores que se han probado.

Los clasificadores explorados han sido Naive Bayes, Random Forest, Support Vector Machines (SVM) y K Vecinos más cercanos. Todos ellos han sido sometidos a una búsqueda exhaustiva de sus hiper-parámetros con el fin de ajustar las mejores condiciones y así conseguir el mejor rendimiento.

A pesar de todas las limitaciones en la dificultad de encontrar información referida al acoso, tras llevar a cabo un proceso manual de etiquetado, tratar los datos utilizando técnicas de procesamiento del lenguaje natural y realizar una serie de pruebas, variantes y combinaciones de distintos algoritmos, los resultados muestran que el mejor clasificador en este caso es SVM con el método de Bag of Words (Bolsa de Palabras) para conversión a vectores, alcanzando una **exhaustividad** en la clasificación de más del 73%, y una **precisión** del casi 64%.

Abstract

Bullying is an increasingly visible problem in school classrooms, and concerns experts from all fields. The serious consequences that it generates to the victims suggests the need to study it in order to avoid new cases.

In addition, the use of technologies and social networks by young people and teenagers is growing continuously. This new way of communication has led to the transition from school *bullying* to the joint practice of traditional *bullying* along with that through technological means, known as *cyberbullying*.

The main objective of this Bachelor Thesis is to develop a supervised classifier capable to decide whether a text is classified as *bullying*. In order to reach this goal, I have worked with a set of documents obtained from Twitter, which have been annotated firstly. Afterwards, these tweets have gone through a process in which they have been cleaned and treated, and emoticons, user names and retweets have been eliminated. I have also grouped sets of terms with the same meaning to get a more generic study. Later, I have applied some procedures to convert these processed texts to numerical vectors, so that they can be used as input arguments for the classifiers that have been tested.

The classifiers investigated were Naive Bayes, Random Forest, Support Vector Machines (SVM) and K Nearest Neighbors. For all of them, their hyper-parameters have been adjusted to the best conditions that lead to achieve the best performance.

Despite all the limitations in the difficulty of finding information about *bullying*, dealing with a manual labeling process and the treatise using natural language processing techniques, the best classifier is reached after all the tests, variants and combinations shown in the report. This is SVM with the Bag of Words method for conversion to vectors. The results for this classifier correspond to a 73% of **recall** and almost a 64% of **precision**.

Palabras clave

Acoso, agresores, aprendizaje automático, ciberacoso, clasificador supervisado, máquinas de vectores de soporte, Naive Bayes, Random Forest, redes sociales, SVM, Twitter, vecinos próximos, víctimas.

Keywords

Bullying, bullies, cyberbullying, machine learning, Naive Bayes, nearest neighbors, Random Forest, social networks, supervised classifier, support vector machine, SVM, Twitter, victims.

Agradecimientos

A Rosa y Álvaro, por ayudarme en todo el desarrollo de este trabajo y por sus ideas y motivación.

A mis padres y el resto de mi familia, que me enseñan lo que de verdad es importante, y que me han aguantado durante toda esta carrera.

A mis compañeros de universidad, a los que tengo la suerte de poder llamar amigos. Sobre todo a Blanquiz, que ha sido compañera de prácticas, de penas y de alegrías.

Al resto de mis amigos, con los que el tiempo libre siempre ha sido sinónimo de risas.

A Carlos, que ha sido el mayor apoyo en todo este tiempo y hace que todos los obstáculos sean más fáciles de superar.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE ECUACIONES

INDICE DE TABLAS

1 Introducción

1.1 Motivación, objetivos y limitaciones

Este trabajo pretende contribuir a la detección de un problema cada vez mayor en la sociedad: el acoso escolar o *bullying*. Los casos en España han aumentado en los últimos años y las cifras de víctimas atendidas en la fundación ANAR (Ayuda a Niños y Adolescentes en Riesgo) se han triplicado desde el año 2015. Esto también hace que sea una situación más visible, preocupante y por tanto estudiada y analizada a fondo por expertos y más importante, por padres, profesores y familias.

La interacción a través de las redes sociales es una parte cada vez más imprescindible en la vida de casi cualquier adolescente español. Esta nueva forma de comunicación ha facilitado la aparición del *bullying* en entornos digitales, conocido como *ciberbullying*. Este abre nuevas puertas de acoso a través de mensajes de texto vejatorios, rumores falsos, chantaje, etc. a través de medios de comunicación tecnológicos. Se trata por tanto de un fenómeno relevante debido a las consecuencias graves de tipo psicológico, emocional y de comportamiento que acarrea a las víctimas: baja autoestima, consumo de sustancias, sintomatología depresiva, uso inadecuado o abusivo de Internet, entre otros efectos negativos. La dificultad de abordar, detectar y prevenir el *ciberbullying* supone una limitación y a la vez un reto. En este trabajo se tratará de detectarlo buscando mensajes indicativos de acoso en una de las redes sociales actuales más utilizadas por la gran mayoría de los adolescentes, Twitter, la quinta red social más utilizada en España en 2017.

El objetivo final que se pretende conseguir es detectar *tuits* que puedan ser considerados como acoso dirigido a otro usuario. Para ello, el proceso incluirá la obtención de textos mediante la API de Twitter, la aplicación para usuarios. Estos textos serán tratados siguiendo técnicas de Procesado de Lenguaje Natural (*NLP, Natural Language Processing*), ya que la clave para el estudio es precisamente el contenido de los mensajes.

Por último, se utilizarán algoritmos de *aprendizaje automático* que decidirán si los textos son clasificados como de acoso o no. Parte de este trabajo consiste en encontrar las mejores herramientas y, por consiguiente, los clasificadores que proporcionen los mejores resultados en la detección del acoso a partir del conjunto de datos con el que contaremos.

Un posible obstáculo que se observa inicialmente es la complejidad de encontrar tuits de acoso frente a la mayoría, que no podrán considerarse como tal. Además, pueden confundirse fácilmente con *discurso de odio*, que normalmente está dirigido a un conjunto de personas por su pertenencia a un grupo determinado, mientras que el *bullying* tiene como víctima a una sola persona y se enfoca explícitamente contra ella.

1.2 Organización de la memoria

La memoria de este trabajo consta de los siguientes capítulos:

- **Capítulo 2: Estado del arte**. En esta parte se pondrá en contexto la evolución del *bullying*, y más en concreto del *ciberbullying*. Desde los inicios hasta la actualidad, se comentarán otros estudios sobre este tema y algunos que se han centrado, como este, en la utilización de técnicas de aprendizaje automático para detectarlo en redes sociales.
- **Capítulo 3: Análisis y Diseño**. Se describirán los conceptos base y las decisiones de diseño tomadas para llevar a cabo el trabajo.
- **Capítulo 4: Desarrollo**. En este capítulo se profundizará en cada concepto descrito en el capítulo anterior, describiendo el camino seguido finalmente para conseguir el mejor resultado posible.
- **Capítulo 5: Integración, pruebas y resultados**. Se describirá la integración de todas las fases explicadas en desarrollo, así como las pruebas realizadas y los resultados obtenidos, mostrando todas las variaciones de atributos y parámetros utilizados en los diferentes clasificadores.
- **Capítulo 6: Conclusiones y trabajo futuro**. Se detallarán las conclusiones del trabajo realizado, destacando la elección del mejor clasificador, y se propondrán mejoras para futuras expansiones o adaptaciones.

2 Estado del arte

2.1 Historia del bullying

Este trabajo se centra en la detección del acoso en redes sociales mediante un proceso de clasificación supervisada. Pero primero corresponde explicar qué es acoso correctamente. La definición de acoso trata del hecho de que una persona hostigue, persiga moleste a otra. El verbo acosar refiere a una acción o una conducta que implica generar una incomodidad o disconformidad en el otro [1].

Entre los tipos de acoso, se encuentra el acoso escolar, más comúnmente referenciado como *bullying* actualmente. Es un hecho cada vez más común en las aulas, y es necesario delimitar en qué consiste exactamente. Uno de los primeros en estudiar este comportamiento fue Dan Olweus, al que podemos considerar un pionero en la materia. Él propuso la siguiente definición:

"Un estudiante es acosado o victimizado cuando está expuesto de manera repetitiva a acciones negativas por parte de uno o más estudiantes". (Olweus 1986, 1993)

Estas acciones negativas se refieren a agresiones de tipo físico y/o verbal. Además, para emplear el término *bullying*, esta relación de acoso debe contar con un desequilibrio de poder o de fuerza: las víctimas tienen dificultades para defenderse. Otro aspecto por observar es que es una acción repetitiva, considerándose así cuando ocurre, al menos, 2 o 3 veces al mes [2]. A los agresores se les suele denominar *bullies.*

La creciente progresión de este fenómeno nos obliga a estudiarlo, detectarlo y detener su evolución. Desde el año 2009 se han casi multiplicado por cuatro los casos contrastados y atendidos en España [3], como se puede ver en la Ilustración 1.

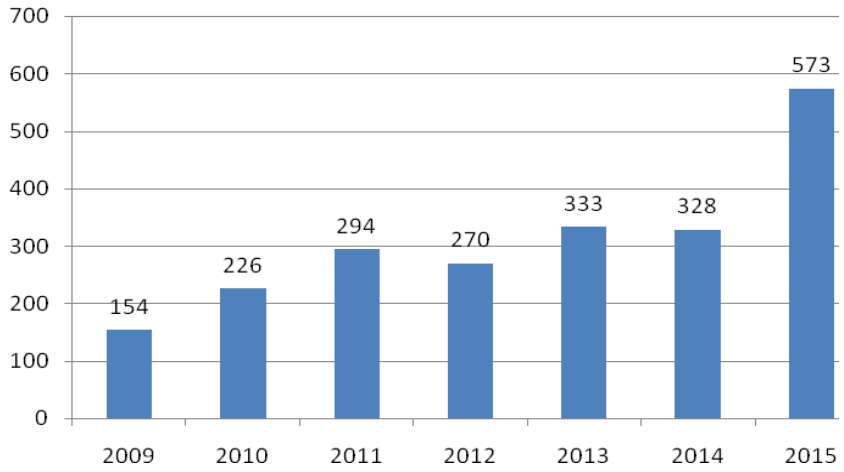

Ilustración 1. Casos de *bullying* en España contrastados y atendidos 2009-2015.

Además, según el estudio que elabora el Observatorio para España de la ONG Internacional *Bullying* Sin Fronteras, se habla de un aumento del 20% anual del 2016 al 2017.

Con los avances de tipo tecnológico en los últimos años, ha aparecido una variante del *bullying*, aquella que se da a través de medios electrónicos como dispositivos móviles, redes sociales o internet: el ciberacoso, al que se conoce también como *ciberbullying*.

2.2 Tipos de bullying

Es obvio que estas agresiones se presentan de múltiples maneras. Basándonos en el Cuestionario de *Bullying* de Olweus (abril 2007), una serie de preguntas que se realizan a estudiantes en un amplio rango de edades, podemos centrarnos en los siguientes tipos:

- Acoso verbal, incluyendo comentarios vejatorios y apodos.
- *Bullying* a través de la exclusión social o aislamiento.
- Acoso físico.
- Amenazas o forzar a otros estudiantes a hacer algo.
- *Bullying* racial.
- *Bullying* sexual.
- *Ciberbullying* (a través de medios electrónicos).

En este caso, nos centraremos en la unión de acoso verbal y *ciberbullying*. Esto significa que el trabajo consistirá en un estudio, mediante técnicas de procesamiento del lenguaje natural, de los mensajes que se escriben en las redes sociales, en concreto Twitter [5].

2.3 Ciberbullying

Las premisas que se han descrito en el apartado 2.1, necesarias para que se produzca el *bullying* "tradicional", también se cumplen con este nuevo formato digital. El desequilibrio entre ambas partes se agrava debido a la anonimidad que existe en las redes sociales, que le aporta al agresor una sensación de poder. Además, no existe el momento cara-a-cara donde el acosador se podría ver influenciado por la respuesta de la víctima, y que en algunos casos le evitaría continuar, por lo que los *bullies* se atreven a decir y hacer cosas que en una situación normal no harían [6].

Una diferencia significativa del *ciberbullying* respecto al acoso en modo tradicional es que se puede realizar 24h al día, 7 días a la semana, lo que hace que la víctima tenga una difícil escapatoria. El hostigamiento es sufrido a diario por casi el 72% de las víctimas. El colegio como escenario habitual cambia a cualquiera imaginable, ya que cualquier dispositivo a través del cual sea posible mandar un mensaje de texto, audio, fotos o vídeos sirve como medio de comunicación. Además, si se considera el alto porcentaje de jóvenes que poseen un teléfono móvil – un 86% según el informe *Sociedad digital en España 2017* – nos podemos hacer una idea de la magnitud del asunto.

Actualmente, el *ciberbullying* representa ya el 25% de los casos de acoso escolar en España [7]. Se observa a través de las redes sociales, sobre todo las de mensajería instantánea. Además de generar nuevas víctimas, es, sobre todo, una forma de agravar el "*bullying* tradicional" descrito anteriormente. Tan sólo un 10-15% de los casos registrados se consideran únicamente ciberacoso, mientras que el resto es una mezcla de ambos tipos de acoso [4].

2.3.1 Tipos de *ciberbullying*

El *ciberbullying* se puede presentar de diferentes modos. Los agresores pueden suplantar la identidad de otra persona, pirateando su cuenta personal y manteniendo conversaciones o publicando información falsa, o en cualquier caso sin permiso del *hackeado.* Otra práctica habitual, y cuyo aumento es el más acentuado es la difusión de información privada. La Ilustración 2 muestra los distintos tipos de manifestaciones del acoso cibernético y su evolución en el año 2016 con respecto a años anteriores.

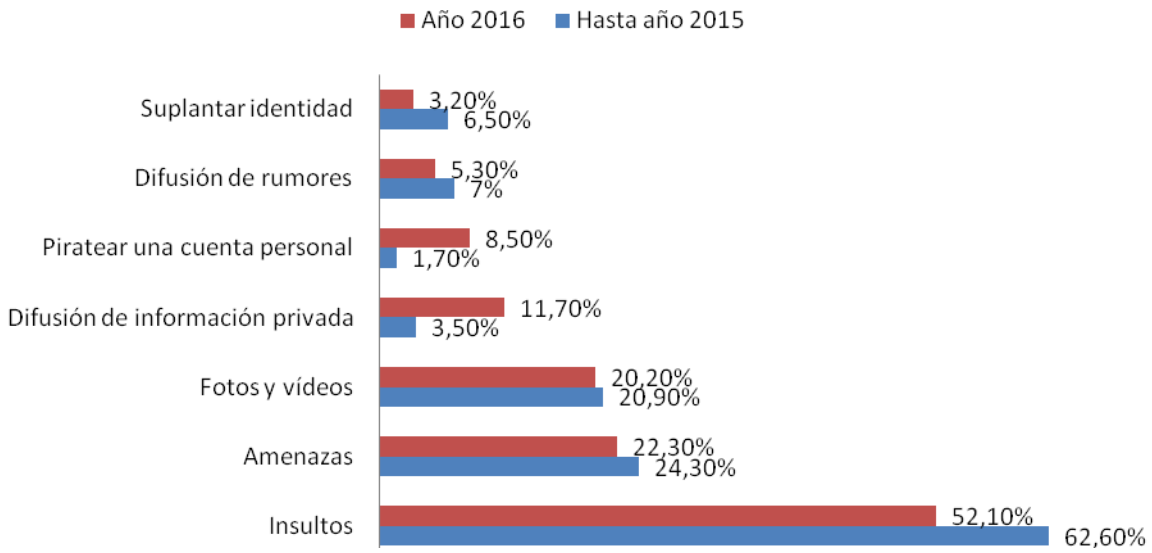

Ilustración 2. Representaciones del *ciberbullying* y su progresión 2015/16.

Como se puede observar, los casos más comunes según este estudio [3], son el insulto (62%) y las amenazas (24%). Es aquí donde centraremos este trabajo. Estudiando los textos de los *tuits* entrenaremos los clasificadores del modo en que más adelante se explicará, para detectar el acoso, que es la finalidad de este trabajo.

2.4 Trabajos en la misma línea

2.4.1 Estudios sobre la evolución del *ciberbullying*

El avance de las tecnologías también ha influido directamente en lo que entendemos hoy por *ciberbullying*, sus limitaciones y su alcance. Cada día la información disponible en Internet y las interacciones que tienen lugar utilizando esta red crecen y es imposible controlar todo lo que sucede *en tiempo real*. Además de utilizar la Web con fines tradicionales relacionados con la consulta de información, hoy en día uno de los usos más extendidos de la Web por parte de los jóvenes consiste en comunicarse mediante redes sociales y de mensajería instantánea.

Este proceso de evolución ha llevado al ciberacoso a un nuevo estado. Este se caracteriza por la rápida **escalabilidad** de los mensajes, que pueden llegar en poco tiempo a un público mucho mayor del que se pretendía en un primer momento; **audiencia invisible** que agrava el problema anterior; y **replicabilidad** o capacidad de reproducción de un mensaje [10].

El hecho de comunicarse a través de una red provoca, en muchas ocasiones, una desinhibición virtual, que permite a los agresores olvidarse de que los mensajes están realmente dirigidos a otras personas, perdiendo la empatía. De la mano, encontramos la **anonimidad** que proporciona la web, lo que dificulta seguirles el rastro a los agresores internautas.

Durante las últimas décadas, el *ciberbullying* ha sido objeto de estudio para muchos investigadores. Se pretende esclarecer las edades más castigadas por este fenómeno, los tipos de personas que son más propicias a agredir a otras, y las posibles repercusiones que tienen estas acciones a lo largo del tiempo.

El perfil que destaca de entre las víctimas del ciberacoso es mujer en un 66% de los casos, de unos 11 años [14]. Por otro lado, los agresores suelen tener una edad similar al de la víctima, y en un 89% de las ocasiones se trata de compañeros de clase.

Es necesario que estos casos tengan una mayor visibilidad, tanto para los profesionales académicos y padres como para los propios alumnos. Los directores de centros escolares antes eran citados un 4% de las ocasiones por los protagonistas del problema, mientras que ahora son mencionados alrededor de un 75% de las veces, incluyéndoles como grupo activo y consciente de la situación [14]. Es importante a su vez, que los profesores sean más consecuentes con estas situaciones violentas, y tomando medidas. Es el camino para reducir las alteraciones psicológicas que conlleva el *bullying*, que se presenta de manera "grave" en el 33% de los casos, y "media" en el 63%.

2.4.2 Estudios que utilizan técnicas de aprendizaje automático

La cada vez mayor visibilidad que se aprecia en el tema del *ciberbullying* hace que expertos de todos los ámbitos quieran ayudar a la pronta detección y conseguir reducir el número de casos. Existen investigaciones que utilizan técnicas informáticas y, en concreto, de aprendizaje automático con este fin. Es más, algunas redes sociales como Instagram ya están incorporando la inteligencia artificial para bloquear automáticamente comentarios que se consideren ofensivos.

Muchos son los conflictos asociados a este proceso: los mensajes ofensivos no tienen por qué contener palabras problemáticas por sí mismas y pueden llevar implícito un sarcasmo casi imposible de localizar mediante técnicas de NLP. La jerga de Internet cambia constantemente, haciendo que las palabras varíen y el vocabulario de tipo conflictivo crezca. Ya existen repositorios de palabras anotadas con puntuaciones negativas y positivas conforme al tipo de reacción o sentimiento que provocan, que pueden ser de utilidad para llevar a cabo un análisis de sentimiento [17].

Las redes sociales de las que los expertos se descargan los conjuntos de datos para estas investigaciones son varias, desde Youtube, Formspring o Twitter hasta MySpace o AskFM [19]. También existe una gran variedad de técnicas de limpieza y tratamiento de los mensajes de texto, que son de gran relevancia para tratar de conseguir buenos resultados. Además, los lenguajes de programación más utilizados son Java y Python, y las herramientas Weka o Scikit.

Cabe destacar que existe un gran número de clasificadores automáticos y que probar cada uno de ellos sobre el conjunto de documentos a tratar sería de un alto coste computacional. Es por eso por lo que cada estudio realiza ensayos sobre los algoritmos que puedan resultar convenientes dependiendo del lenguaje utilizado, la herramienta disponible, u otros factores como, por ejemplo, el etiquetado por parte de personas y la posible subjetividad de los etiquetadores.

Son todas estas variantes de parámetros utilizados, cálculos y cómputos las que hacen de cada estudio algo distinto. Lo que sí que comparten es la determinación para erradicar este problema que cada vez es más visible y preocupante.

3 Análisis y Diseño

En este apartado se describen las ideas iniciales y principales de la solución propuesta para la detección del ciberacoso, así como las decisiones de diseño tomadas durante el proceso, incluyendo las técnicas y algoritmos elegidos para la realización de este trabajo. En la Ilustración 3 se muestra un esquema general del proceso completo desde que se accede a la red social Twitter para obtener los mensajes escritos por los usuarios hasta que se obtienen los resultados de los clasificadores.

Primeramente, los tuits son obtenidos de Twitter. A continuación, son filtrados conforme a una lista de palabras que designan maldad por sí mismas. Una vez que los tuits ya están filtrados, se procede al etiquetado manual. A continuación, se trata y limpia el texto para conseguir una generalización de estos. Ya que los clasificadores necesitan datos numéricos, se convierten los documentos a vectores, consiguiendo la matriz de pesos.

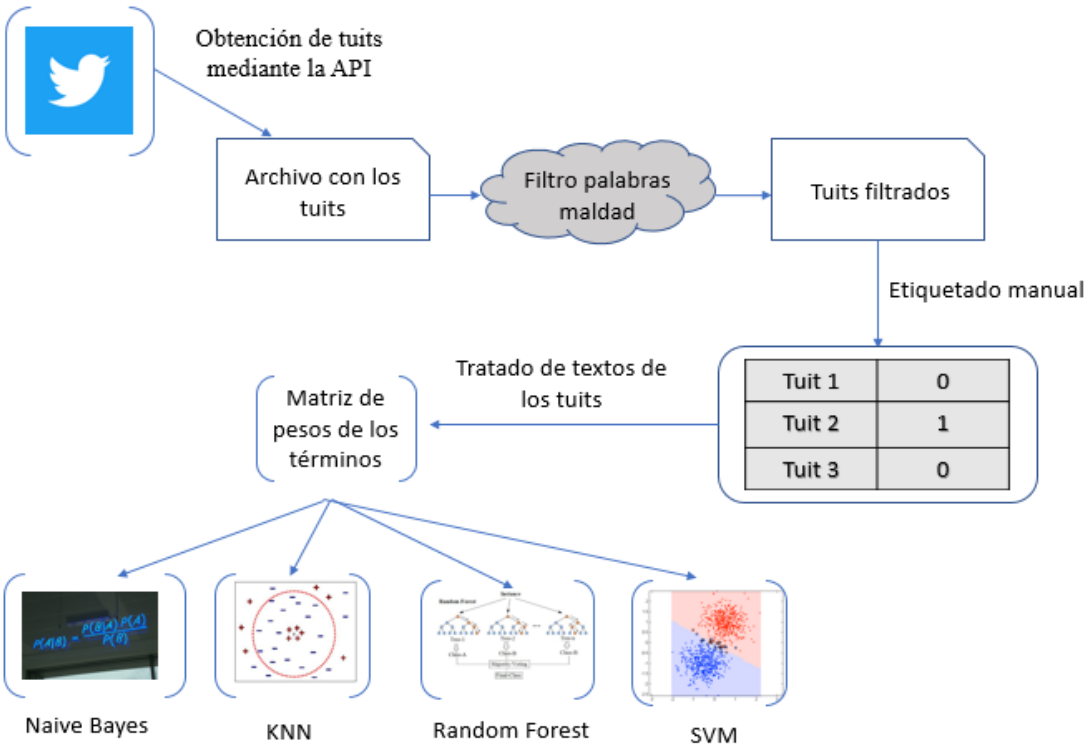

Ilustración 3. Esquema general del proceso.

A continuación, se describen de manera más detallada los requisitos funcionales y no funcionales a satisfacer, así como las distintas partes de este esquema que conforman el proceso.

3.1 Análisis

Los requisitos por satisfacer son:

3.1.1 Requisitos funcionales

- RF1: Clasificación de tuits
  El clasificador podrá recibir como entrada un texto nuevo y devolverá 1 si es clasificado como un mensaje de acoso y 0 en el caso contrario.
- RF2: Elección de parámetros
  Se podrá introducir como argumento de entrada un **archivo** csv donde se encuentren los **textos clasificados**, con el fin de poder aumentar el conjunto o variar el criterio de etiquetado.

3.1.2 Requisitos no funcionales

3.1.2.1 Portabilidad

- RNF3: El clasificador se podrá utilizar en sistemas con Python 3 instalado.

3.1.2.2 Usabilidad

- RNF4: La utilización del clasificador será fácil para el usuario final, y se incluirá un manual donde se explicarán los aspectos más relevantes del proceso.

3.1.2.3 Rendimiento

- RNF5: El rendimiento del clasificador será superior al rendimiento de un clasificador aleatorio, y también tendrá mayor precisión que un clasificador que identifique todo como 'no *bullying'*.

3.2 Diseño

3.2.1 Obtención de tuits

El trabajo se centra en detectar el *ciberbullying* en la red social de Twitter. Existen otras muchas redes sociales de las que se podrían extraer mensajes entre usuarios, algunas mencionadas en el Apartado 2.4.2. Podría ser de utilidad conseguir mensajes de alguna aplicación de mensajería instantánea como Whatsapp o Telegram. En ellas, los adolescentes, en chats grupales o individuales, son más propensos a enviar mensajes ofensivos. Sin embargo, la obtención de datos de este tipo de aplicaciones no es sencillo, pues se considera actuar en contra de la privacidad de los usuarios. Muchas de ellas no tienen APIs gratuitas por esta misma razón. Por otra parte, según un estudio llevado a cabo en 2017, las tres redes sociales en las que se detectaron más casos de *ciberbullying* fueron Instagram, Facebook y Snapchat [20]. La Ilustración 4 muestra los porcentajes de adolescentes en cada red social junto con la proporción de ellos que han sufrido casos de *ciberbullying.*

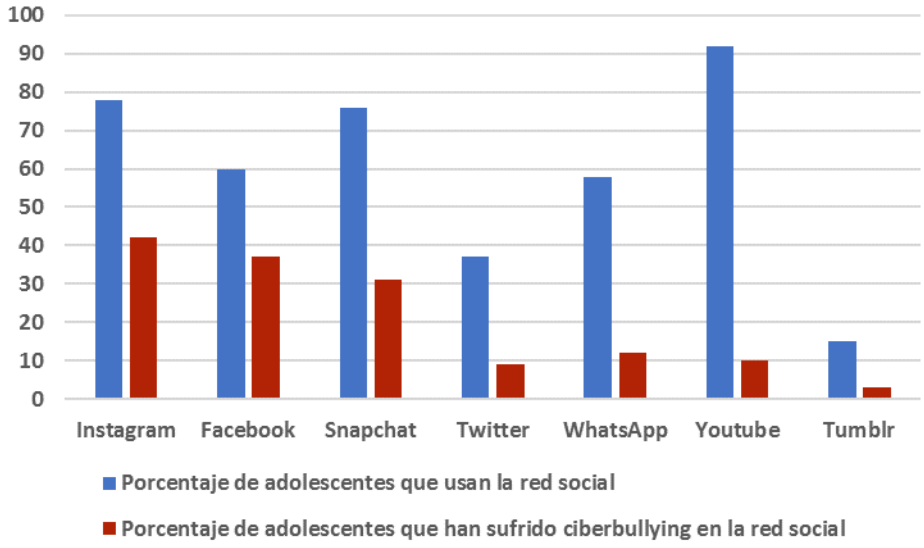

Ilustración 4. Comparativa por red social de *ciberbullying* en 2017.

En nuestro trabajo, se ha elegido Twitter porque tiene una aplicación libre y gratuita para acceder a los mensajes que en ella se escriben. Además, recoge gran cantidad de los mensajes agresivos que han sido observados a figuras públicas en las redes sociales [28]. Los tuits que se usarán en la fase de entrenamiento son obtenidos mediante la API de Twitter de búsqueda por palabra clave. Esta devuelve un archivo JSON con toda la información que esta plataforma recoge. De todos los campos disponibles, se seleccionarán para el estudio el autor, la fecha y el texto completo. A partir de este último dato se seguirá el esquema mostrado en la Ilustración 3.

3.2.1.1 Palabras más comunes y segunda búsqueda

Estudiando las palabras más comunes de los textos de los tuits sería posible realizar una segunda búsqueda en la API de Twitter para encontrar más mensajes que conlleven *bullying*. Se repetirían las búsquedas en la API mediante palabra clave, pero en este caso, con los términos más comunes que se han encontrado al estudiar el primer corpus que se obtuvo. De esta manera, si el conjunto de datos del que se dispone en un primer momento no se considera lo suficientemente grande, se ampliaría para el entrenamiento de los algoritmos que se presentan a continuación.

3.2.2 Filtrado de tuits

No todos los tuits extraídos serán interesantes de cara al problema de localizar ciberacoso. Se aprecia la necesidad de filtrar, por tanto, los mensajes obtenidos, y seleccionar aquellos que contengan palabras que impliquen sentimientos negativos: odio, envidia o desprecio.

3.2.3 Limpieza texto

Para que todos los mensajes puedan ser aceptados por el clasificador, primero se necesita tratar los textos: eliminar emoticonos, tener en cuenta signos de puntuación y exclamación,

menciones a usuarios, y links a otras páginas web. Esto se ha realizado mediante un proceso de separación de elementos de cada mensaje (*tokenización*) e identificación de estos componentes mediante expresiones regulares.

Además, una técnica básica en el tratamiento de los documentos de texto es reducir las formas conjugadas de los verbos, y asignar una única representación para las variables de género y número en sustantivos, determinantes y adjetivos, con el fin de llegar a la *forma base* de los términos. Existen dos posibilidades para llevar a cabo este proceso: lematización y *stemming*.

3.2.3.1 Lematización

Los algoritmos de lematización tienen en cuenta el análisis morfológico de las palabras. Se fijan en diccionarios detallados para devolver los términos a su forma base, que, en este caso, se denomina *lema*. Por ejemplo, tal y como se muestra en la Tabla 1, la palabra 'Niñas' tiene 'Niño' como lema, y 'Niñez' a ella misma.

Palabra	Lema
Niñas	Niño
Niñez	Niñez

Tabla 1. Ejemplo de lematización.

3.2.3.2 Stemming

Por el contrario, el proceso de stemming funciona *recortando* el inicio o fin de la palabra, considerando una lista de prefijos y sufijos comunes. Puede funcionar bien en algunos casos, pero presenta limitaciones en la mayoría de ellos, ya que, al eliminar una parte del término, también está eliminando detalles que pueden ser importantes. La Tabla 2 muestra que, al contrario que la lematización, que se fija en su significado morfológico, el stemming decide que 'Niñez' y 'Niñas' provienen del mismo *stem.*

Palabra	Stem
Niñas	Niñ
Niñez	Niñ

Tabla 2. Ejemplo de stemming.

En este trabajo se utilizará la técnica de lematización para la limpieza de los mensajes, al considerar su proceso más acorde y conveniente a nuestro problema.

3.2.4 Etiquetado de tuits

Una vez que los tuits han sido filtrados y tratados, hay que hacer un etiquetado manual. Basándose en las decisiones que los etiquetadores tomen, los clasificadores trabajarán para tomar futuras resoluciones.
Lo ideal sería tener un grupo de etiquetadores. Cada persona puede ser en cierta manera parcial o subjetiva en sus clasificaciones, que se pueden ver influidas por aspectos tales

como su estado de ánimo/humor o cansancio en el momento de realizar el etiquetado, o su forma de entender las palabras, el acoso cibernético y la ironía.
Pero la subjetividad individual se lograría disminuir si finalmente se escoge la etiqueta mayoritariamente asignada. Si, por el contrario, se cuenta con un solo etiquetador, el riesgo de que la subjetividad influya en los resultados es mayor.

3.2.5 Conversión a vectores

Una vez que los tuits han sido etiquetados, es necesario convertirlos a un formato adecuado para poder ser utilizados como entrada de los algoritmos de aprendizaje. Los clasificadores necesitan una representación numérica de cada tuit. Una forma de conseguir esa representación es generar una matriz numérica de documentos que incluya los pesos de cada término para cada documento. Este cambio de información textual a numérica se denomina vectorización y puede ser realizado de varias maneras. A continuación, explicaremos tres de ellas.

3.2.5.1 Bag of Words (BoW) o Bolsa de Palabras

Después del proceso de etiquetado de cada mensaje, se crea un diccionario de términos común para todo el corpus. Las palabras pertenecientes a él son todas las que aparecen en el conjunto de tuits, conseguidas tras la tokenización.
Los vectores documento se formarán sencillamente con la frecuencia de cada término en el documento.

3.2.5.2 TF-IDF

Es una técnica de recuperación de la información que da pesos a las palabras que aparecen en los textos, indicando la importancia del término en el documento y con respecto al conjunto completo de tuits.

- Term Frequency o TF. Hace referencia a la frecuencia de un término dentro del documento. Sea *t* el término que se está estudiando y *d* el documento en el que aparece:

$$TF = \frac{Veces\ que\ aparece\ t\ en\ d}{Nº\ total\ de\ palabras\ en\ d}$$

Ecuación 1. TF.

- Inverse Document Frequency o IDF. Expresa lo significante que se considera el término *t* dentro de todo el corpus:

$$IDF = log\left(\frac{Nº\ total\ de\ documentos}{Nº\ de\ documentos\ que\ contienen\ t}\right)$$

Ecuación 2. IDF.

Así, el peso de un término *t* en el documento *d* será, de acuerdo con las ecuaciones Ecuación 1 y Ecuación 2:

$$Peso(t,d) = w(t,d) = TF * IDF$$

Ecuación 3. Peso de un término *t* en el documento *d.*

3.2.5.3 Doc2Vec

Para comprender el funcionamiento de doc2vec, es necesario entender primero **word2vec.** El algoritmo word2vec transforma palabras a vectores que capturen relaciones entre términos, como, por ejemplo, sinónimos, antónimos o analogías. La Ilustración 5 muestra un ejemplo de la relación entre las palabras "man" y "woman" y de la relación entre las palabras "king" y "queen", representadas ambas mediante vectores.

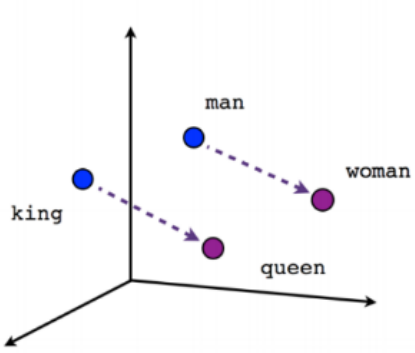

Ilustración 5. Ejemplo de relación entre palabras en word2vec.

La representación word2vec se crea a partir de otros dos sub-algoritmos: el modelo CBoW (Continuous Bag-of-Words) y Skip-Gram. El primero trata de averiguar la palabra por contexto – el resto de las palabras que la acompañan. Se puede apreciar en la Ilustración 6, donde tras recibir los *tokens* "¿", "qué" y "tal", predice que la siguiente palabra será "estás":

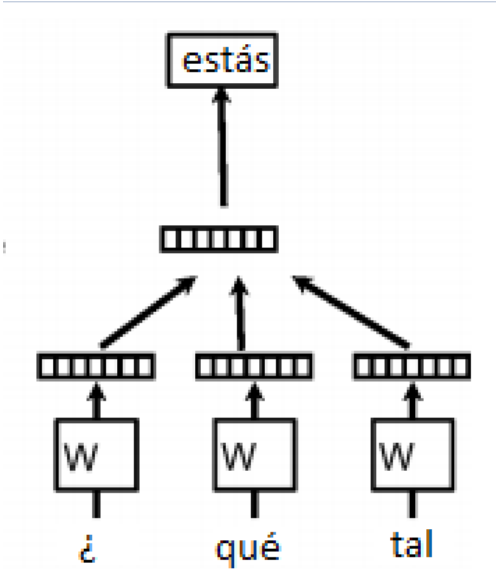

Ilustración 6. Ejemplo de CBoW.

Skip-gram hace lo contrario: en lugar de predecir una palabra en función del contexto, intenta predecir el contexto a partir de una sola palabra.
Por su parte, doc2vec es una extensión de word2vec en la que, aparte de los datos que se tenían en cuenta antes, se incluye un ID del documento en cuestión.
El proceso en sí actúa como una memoria que recuerda "lo que falta" en el contexto del mensaje, o su contexto propiamente dicho. Esto será de utilidad para clasificación de textos ya que se intenta representar el concepto del documento, que se relaciona directamente con la etiqueta de dicho mensaje. En la Ilustración 7, se observa el funcionamiento de Doc2vec, donde se aprecia que es básicamente word2vec junto con el identificador del documento.

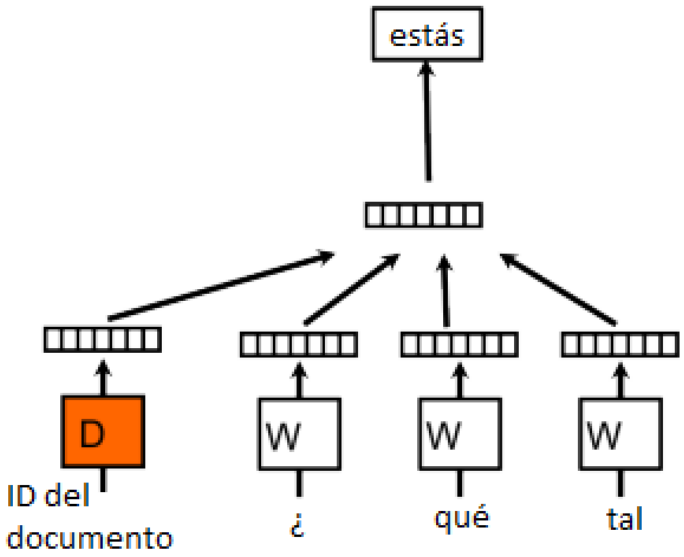

Ilustración 7. Ejemplo de Doc2Vec.

En este trabajo, se aplicarán las dos primeras opciones, y la utilización de Doc2Vec se reservará para el capítulo de anexos.

3.2.6 Clasificadores

Hay variedad de clasificadores de aprendizaje automático que se pueden utilizar para los problemas de clasificación supervisada. Se ha decidido hacer una comparativa durante todo el trabajo entre los cuatro elegidos incluyendo sus propias variantes: Naive Bayes, máquinas de vectores de soporte o SVM, Random Forest y K Vecinos Próximos. Seguidamente, se describe cada uno de ellos y cómo se utilizarán en este trabajo.

3.2.6.1 Naive Bayes

Es un clasificador probabilístico que se basa en el teorema de Bayes. Este plantea:

Sea $\{A_1, A_2, \dots, A_i, \dots, A_n\}$ un conjunto de sucesos mutuamente excluyentes, y tales que la probabilidad de cada uno de ellos es distinta de cero. Sea B un suceso cualquiera del que se conocen las probabilidades condicionales $P(B|A_i)$. Entonces, la probabilidad $P(A_i|B)$ viene dada por la expresión:

$$P(A_i\,|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

Ecuación 4. Fórmula general del Teorema de Bayes

En la Ecuación 4:

- $P(A_i)$ son las probabilidades a priori
- $P(B|A_i)$ son las probabilidades de B condicionadas al suceso A_i
- $P(A_i|B)$ son las probabilidades a posteriori
- $P(B)$ es la probabilidad de que ocurra el suceso B independientemente de las hipótesis, siendo calculado como:

$$P(B) = \sum_k P(B|A_k)P(A_k)$$

Ecuación 5. Probabilidad de un suceso.

En el caso de este trabajo, se trata de predecir la probabilidad de que un nuevo documento pertenezca a una clase $P(A_i \mid B)$ a partir de la probabilidad de los documentos dada la clase $P(B|A_i)$ y la probabilidad a priori de la clase en el conjunto de entrenamiento $P(A_i)$.

Cabe destacar que este clasificador se plantea sobre una situación nada realista (*naive*): que todos los atributos en los que se basa son independientes. A pesar de esto, consigue un alto rendimiento y unos buenos resultados en la mayoría de los casos de clasificación de texto y procesamiento del lenguaje natural.

Este clasificador presenta tres variantes: Gaussiano, Bernoulli y Multinomial. Las pruebas correspondientes se presentarán en la parte correspondiente de esta memoria.

A pesar de que el tradicional algoritmo toma como entrada una matriz de frecuencias, donde cada documento viene representado por la cantidad de veces que sus términos aparecen en él (como el método BoW), también recoge buenos resultados si introducimos para el entrenamiento el resultado de utilizar TF-IDF [21].

3.2.6.2 Support Vector Machines (SVM)

Una *máquina de vector soporte* es un clasificador de aprendizaje supervisado. Se basa en un conjunto de muestras de entrenamiento ya etiquetadas, para construir un modelo que prediga la clase de las muestras nuevas. Este algoritmo representa las muestras como puntos en el espacio, y separa las diferentes clases mediante un hiperplano. La característica de este hiperplano es que se trata del óptimo: el que tenga la máxima distancia (margen) con los puntos que estén más cerca de él mismo. A la hora de definirlo, sólo se consideran los ejemplos de entrenamiento de cada clase que caen justo en la frontera de dichos márgenes. Estos ejemplos reciben el nombre de *vectores soporte*.

Los problemas que abordan las SVMs corresponden a problemas de optimización con restricciones lineales. Haciendo uso de la *Teoría de la Optimización* se pueden resolver.

Para un ejemplo en el que los documentos consten de dos atributos, el hiperplano será una línea dividiendo un plano en dos partes, donde cada clase esté a un lado. En laIlustración 8, las líneas discontinuas también serán planos separadores, pero el óptimo se corresponde con la continua, es decir, el que consigue el máximo margen con los puntos más cercanos a él.

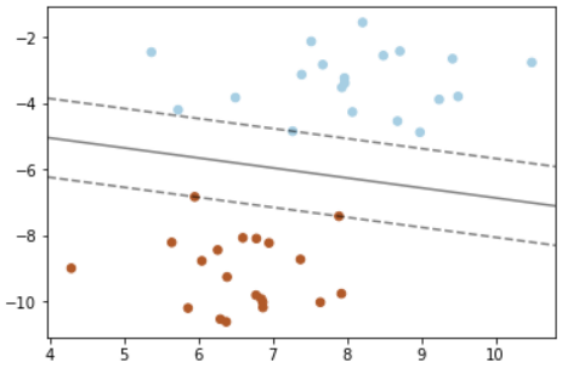

Ilustración 8. Ejemplo de hiperplano separador óptimo de dos clases.

Si las clases constaran de tres atributos, los textos se representarían como puntos en un espacio tridimensional. Por tanto, el hiperplano separador óptimo será un plano como el mostrado en la Ilustración 9.

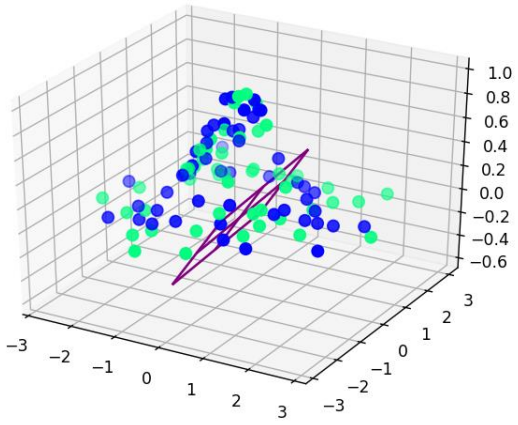

Ilustración 9. Ejemplo de hiperplano separador óptimo de dos clases.

En nuestro trabajo, se trabajará con un hiperplano r-dimensional, donde *r* corresponderá a la cantidad de atributos que se tienen en cuenta para el etiquetado de los textos. Es decir, será el tamaño de los vectores con los que se representa cada documento.

Es evidente que no todos los problemas que aborde el algoritmo serán linealmente separables, como es el caso de la Ilustración 8. En la mayoría de las ocasiones nos encontraremos con problemas no separables, como el mostrado en la Ilustración 10.

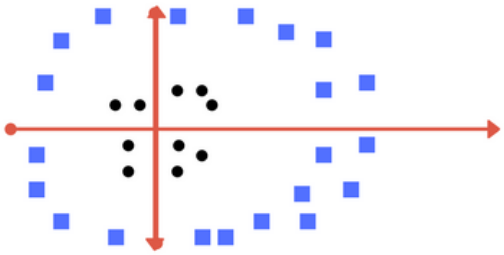

Ilustración 10. Ejemplo no separable de dos clases.

Las clases mostradas en la Ilustración 10 no pueden ser separables mediante una línea. Es necesario aplicar una transformación y añadir una dimensión más, *el eje z*.
Si consideramos la transformación $w = x^2 + y^2$ y representamos este nuevo eje, obtendremos una separación lineal (ver Ilustración 11).

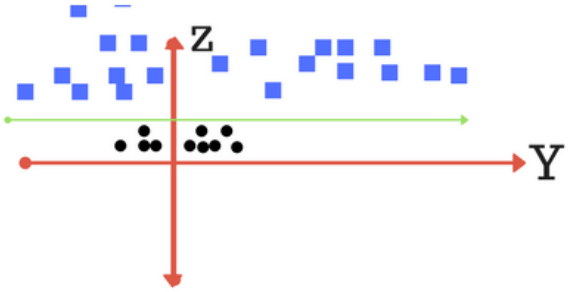

Ilustración 11. Ejemplo anterior tras la transformación.

Si ahora deshacemos el cambio realizado al plano original, la separación de las clases se ha conseguido mediante una circunferencia. Es lo que se puede observar en la Ilustración 12 y se denomina *kernel.*

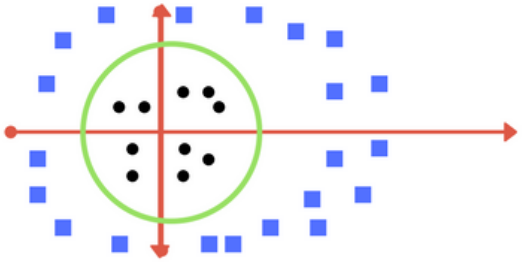

Ilustración 12. Ejemplo anterior tras deshacer el cambio al plano orginal.

Además, puede que las clases se entremezclen, y dependerá de los parámetros de entrada del algoritmo el procedimiento a seguir. Una opción es separar totalmente las clases sin dejar ningún punto al otro lado de la línea que define el hiperplano separador (RBF). Esto conlleva mucho tiempo de computación. Por otro lado, es posible dejar algunos puntos que no estén en su clase correspondiente (lineal), con el fin de agilizar el proceso y ya que no resta demasiada precisión al modelo. Esta comparativa se muestra en la Ilustración 13.

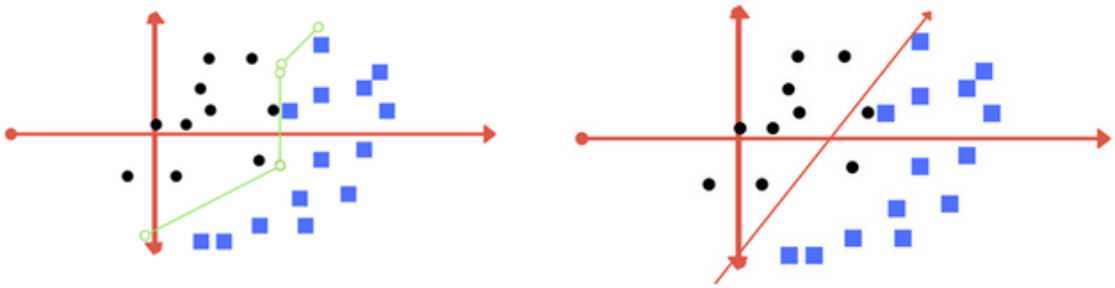

Ilustración 13. Diferencia entre kernel RBF (Radial basis function) y lineal.

3.2.6.3 K Vecinos más próximos (k Nearest Neighbors, kNN)

Este clasificador basa sus predicciones en las etiquetas de los *k* vecinos más próximos. Lo habitual es que esta distancia entre documentos se calcule mediante la distancia euclídea:

$$D(p,q) = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Ecuación 6. Distancia euclídea.

Donde *p* y *q* son dos textos y cada p_k, q_k los atributos o características para comparar entre ambos.
El parámetro *k* en kNN definirá el número de vecinos que se tendrán en cuenta a la hora de decidir a qué clase pertenece un nuevo documento. Por ejemplo, en la Ilustración 14, el 'documento estrella' sería clasificado como perteneciente a la clase B si *k=3*. En cambio, si *k=6*, éste sería etiquetado como clase A:

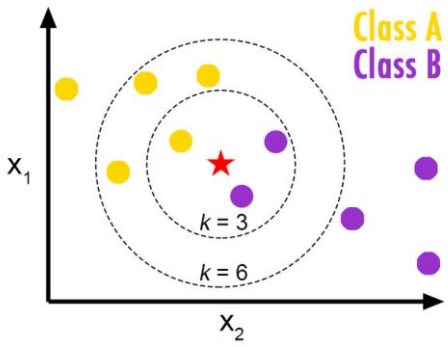

Ilustración 14. Ejemplo de clasificación por el algoritmo de k vecinos más próximos.

3.2.6.4 Random Forest

El clasificador *Random Forest* combina el funcionamiento de varios algoritmos. Crea un conjunto de árboles de decisión a partir de dividir de forma aleatoria (*random*) el total de los textos que se usan en la fase de entrenamiento. Después, teniendo en cuenta todos estos árboles de decisión, el resultado final es lo que la mayoría haya predicho. El comportamiento de este clasificador obtiene un buen rendimiento. Se debe a que un número pequeño de árboles de decisión puede derivar en *ruido*[1], pero al tener en cuenta una gran cantidad de ellos, se consigue reducir y llegar a un resultado preciso.

[1] Efectos negativos en la clasificación relacionados con datos indeseados.

4 Desarrollo

4.1 Descarga de tuits

Con el fin de encontrar un conjunto de *tuits* cuyos textos sirvan de muestra para este problema de detección del *bullying*, se extraen mensajes de la red social Twitter mediante su API. Para ello se crean seis aplicaciones que consiguen optimizar el proceso, ya que las llamadas a esta API cuentan con restricciones. Dependen del tipo de autenticación; en nuestro caso, respecto a la API de búsqueda, se cuenta con un máximo de 180 llamadas cada 15 minutos y por cada aplicación [8].
En estas llamadas a la red social la búsqueda se realiza por palabra clave. Contamos con una lista de 98 términos considerados comunes en mensajes de acoso en redes sociales referentes a raza, género, religión y orientación sexual. Se obtiene el autor, fecha y texto del *tuit* y se consiguen 48.150 mensajes en total. No se descargan los retuits, para evitar repeticiones del mismo mensaje, y sólo se seleccionan los mensajes que contengan mención a otro usuario de la red social - especificado mediante "@", para forzar a que el mensaje vaya dirigido a otra persona.
Cabe destacar que todo el proceso de código se ha realizado mediante el lenguaje de programación Python, en la herramienta Jupyter Notebook y usando librerías que se irán citando a lo largo de la memoria.

4.1.1 Volcado a archivo CSV

Una vez obtenidos los casi 50.000 tuits, se decide no realizar una segunda búsqueda ya que se cuenta con un corpus relativamente amplio. La mayoría de los tuits son devueltos por la API incompletos, pero con un link que nos redirecciona al tuit en la página. Accedemos a ellos para contar con el texto completo y conseguir el contexto total del mensaje en cuestión.

4.2 Filtrado de tuits

Conviene filtrar los más de 48.000 tuits conseguidos en las búsquedas mediante la API de la red social. Obligamos a que los tuits deban contener también alguna de las palabras del fichero *palabrasFiltro.txt*. Son términos despectivos por ellos mismos, por lo que, si acompañan a las palabras previas, es más probable que el mensaje en cuestión sea *bullying*. Por ejemplo, no todos los mensajes que contengan la palabra "fascista", que se encuentra en la lista de palabras clave para la búsqueda, serán considerados como acoso. En la Tabla 3 se observan un par de ejemplos. La columna de la izquierda contiene tuits que no han pasado el filtro, porque a pesar de ser obtenidos en la búsqueda por palabra clave ("fascista" en este caso), no contienen ningún término de la lista del filtro. Por el contrario, la columna derecha muestra tuits interesantes para el estudio, ya que han sido elegidos tras el filtrado.

Tuits que no pasan el filtro	Tuits que se mantienen después del filtrado
"@publico_es Lo raro no es que alguien de Fuerza Nueva diga esto. Lo intolerable es que se le de voz a un fascista como @Tebasjavier"	"@Carmenprofesor1 @AdrianBlanco6 @MomentsES Te metes tu por la boca tu mierda fascista basura facha."
"El Sindicato de Estudiantes de la UMH denuncia una agresion fascista via @La_SER"	Señores fascistas de @laliga peleles del facha @Tebasjavier, puto titere rastrero del Nuevo Caudillo Don Florleone... No pueden dar más asco...

Tabla 3. Ejemplo de filtrado de tuits.

Al finalizar el proceso de filtrado, contamos con un total de 7508 tuits.

4.3 Etiquetado

Una vez que todos los mensajes que tenemos pueden ser objeto de estudio para el acoso, el proceso de etiquetado es manual. Se trata de etiquetar los mensajes con '0' si no son considerados como acoso por el etiquetador, y, por el contrario, con '1' si el texto conlleva *bullying*. El proceso ideal de contar con varios etiquetadores comentado en 3.2.4 no ha podido ser llevado a cabo, por lo que las decisiones futuras de clasificación se verán influidas por lo que el único etiquetador haya considerado durante el proceso de etiquetado.
Tras este proceso, se cuenta con 3053 tuits considerados de acoso y 4455 que no reflejan acoso. Como curiosidad, cabe comentar que inicialmente se pensaba que el resultado sería más desbalanceado, pero el conjunto de datos disponible muestra que no es así.

4.4 Limpieza de los textos

Llegados al punto en el que contamos con todo nuestro corpus etiquetado, es momento de limpiar el texto para aplicar un procesamiento del lenguaje adecuado.
Previamente al proceso de filtrado, ya se había realizado un primer tratamiento de los textos que resulta fundamental: la *lematización.* Este procedimiento se realizó antes de filtrar los tuits, ya que los verbos de la lista se encuentran en infinitivo, los adjetivos y sustantivos en singular, etc.

Hasta hace relativamente poco tiempo, la mayoría de los lematizadores trataban únicamente textos en inglés. Era por tanto necesario traducir el texto a este idioma, aplicar el algoritmo en cuestión, y deshacer la traducción. En este trabajo se ha elegido el lematizador TreeTagger [26], que tiene un paquete ya definido en Python. Su última versión, añade la funcionalidad del lenguaje español, lo que nos permite ahorrarnos el paso de traducción.

Asimismo, se ha procedido a una decodificación de los textos, que se encontraban codificados en utf-8.

Es hora de proceder con el resto de los métodos de limpieza:

- Se define una lista de stopwords. También se las suele denominar palabras vacías, ya que carecen de significado por sí mismas. Estas palabras serán eliminadas de los mensajes. Suelen ser determinantes, preposiciones, artículos y conjunciones. Además, su frecuencia en los textos es tan alta que pierden relevancia en el análisis de los textos. Aparte de las palabras vacías del castellano, añadimos a la lista los símbolos de puntuación.
- Mediante la detección de expresiones regulares se identifican cadenas de más de dos caracteres de símbolos de exclamación, interrogación, etc. Son identificadas como "EXPR".
- Las menciones a otros usuarios de Twitter son reemplazadas por "USER", y los links a otras páginas web por "URL". Los emoticonos son detectados según su código *unicode* y posteriormente eliminados.
- Todas las posibles cadenas que expresan risas o carcajadas son unificadas en "RISAS".
- Por último, se eliminan los espacios múltiples.

4.5 Conversión a vectores

Para convertir los textos a vectores de modo que puedan ser aceptados como entradas por los algoritmos de aprendizaje, se ha utilizado tanto BoW, como TF-IDF, *(Term Frequency-Inverse Document Frequency).*

En el siguiente capítulo se mostrarán resultados obtenidos con ambos métodos. Si bien la metodología de Bolsa de Palabras parece a primera vista más sencilla y TF-IDF más completa, todo depende del conjunto de datos y del clasificador, por lo que se realizarán todas las pruebas necesarias para determinar cuáles dan lugar a los mejores resultados.

Los pasos seguidos en la limpieza de los textos (Apartado 4.4) - la eliminación de las *stopwords*¸ el reemplazo de distintas expresiones con mismo significado por un único término, así como todos los usuarios por la palabra "USER" – cobran sentido al utilizar el algoritmo TF-IDF. Al agrupar términos en expresiones generales y eliminar otros, conseguimos darle el peso adecuado a cada atributo.

Los clasificadores tomarán como entrada, por tanto, una matriz de pesos, donde las filas serán los tuits, y las columnas se corresponderán con los n-gramas. Se ha decidido crear las matrices con unigramas, o sea, términos, y bigramas (n-grama con n=2), un conjunto de dos palabras consecutivas que se tratarán como una sola. Con ello se pretende estudiar si en estos textos las expresiones más comunes son palabras únicas o se manifiestan en mayor número las "frases hechas" compuestas por dos términos.

La Ilustración 15. Diferencia entre unigramas y bigramas.Ilustración 15 muestra una frase y cómo se puede aplicar tanto el uso de unigramas como de bigramas a la hora de tokenizar.

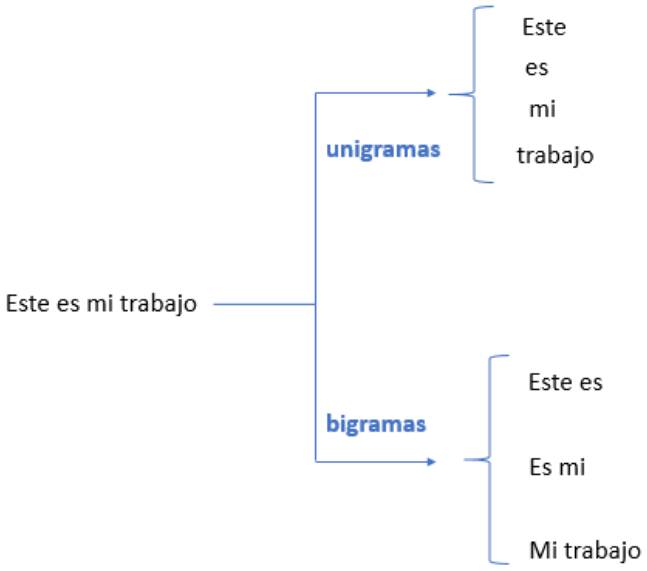

Ilustración 15. Diferencia entre unigramas y bigramas.

4.6 Clasificadores

Se realizan pruebas con los cuatro clasificadores propuestos en 3.2.6, ya que se consideran de los mejores para resolver problemas de clasificación binaria de textos. Todos ellos tienen implementaciones en Scikit-learn de Python, que es la librería de aprendizaje automático que se ha utilizado. Es de software libre y específica para abordar problemas de clasificación, regresión y algoritmos de clustering.

Cada uno de los métodos de declaración de los clasificadores requiere unos parámetros de entrada que se han ido ajustando para mejorar el rendimiento y los resultados. Es más, la librería utilizada incluye una función (*GridSearchCV*) que se encarga de hacer una búsqueda exhaustiva, entrenando los clasificadores con varios valores para los parámetros elegidos. Al final del proceso, devuelve las mejores valoraciones para cada uno de estos hiper-parámetros, basando su criterio en la precisión y en el conjunto de datos de entrenamiento.

Para el proceso de entrenamiento (*training set*), se considera las 2/3 partes del conjunto total de tuits ya etiquetados. La otra tercera parte (*test set*) servirá para asegurar que funciona correctamente.

A continuación, se comentan los parámetros de entrada de cada clasificador.

4.6.1 Naive Bayes

Se han probado tanto Multinomial (*MultinomialNB*) como Bernoulli (*BernoulliNB*). La diferencia entre ellos es que la variante Multinomial considera la frecuencia de aparición de cada término en los documentos en lugar de una ocurrencia binaria. En ambos casos los parámetros de entrada que se probarán mediante el método GridSearch sobre el conjunto de entrenamiento son los mismos:

- **Alpha**: es un hiper-parámetro de este algoritmo que se relaciona con el *Laplace smoothing*. Es una técnica para evitar las probabilidades nulas que aparecen con los n-gramas nunca vistos por un clasificador. Para activar esta función Alpha tiene que ser 1, y así es como la especificamos.
- **Fit_prior**: indica la capacidad de aprender probabilidades de las clases anteriores o no. Puede ser, por tanto, *True* o *False*.
- **Class_prior**: mediante este parámetro se especifica la probabilidad previa de las dos clases presentes en el conjunto, es decir, *bullying* o no, antes de entrenar el modelo. Se trata de definir los pesos de las clases, por ejemplo, si no están balanceadas y queremos hacer que lo estén. Se ha definido la lista [[0.4, 0.6], [0.35, 0.65], [0.65, 0.35], [0.6, 0.4], [0.55, 0.45], [0.3, 0.7], [0.7, 0.3]], donde cada sub-lista corresponde al peso correspondiente a cada clase.

4.6.2 SVM

La clase referida a este clasificador de Sklearn es SVC (*Support Vector Classification*). Su implementación está basada en otra librería, *libsvm*. Los parámetros analizados por GridSearch en este caso son:

- **Kernel**: Especifica el tipo de núcleo que el algoritmo usa. En la lista hemos definido que puede tomar dos valores, lineal y RBF, mostrados en la Ilustración 13 en el capítulo 163.2.6.2.
- **C**: es un parámetro que indica a la optimización de SVM cuánto se desea evitar la clasificación errónea de cada ejemplo de entrenamiento. Para valores grandes de C, el algoritmo escoge un hiperplano con un margen más pequeño si éste realiza un trabajo mejor clasificando todos los tuits de entrenamiento. Por el contrario, un valor pequeño de C buscará un margen más amplio para el hiperplano separador, aunque haya más puntos clasificados erróneamente.
  Probaremos con una secuencia de potencias crecientes [27] de 2: $C = [2^{-5}, 2^{-4}, 2^{-3}, \dots, 2^{2}, 2^{3}]$
- **Gamma (sólo presente con kernel RBF):** técnicamente, es la función inversa a la variación estándar del núcleo RBF (función Gaussiana), usada como medida de similitud entre dos puntos. Un valor bajo de gamma define una función Gaussiana con una varianza alta. Así que, en este caso, dos tuits podrían considerarse similares, aunque estén bastante separados en el plano. Un valor alto del parámetro definirá una función con baja varianza, y los tuits se considerarán similares sólo si están cerca el uno del otro. Al igual que C [27], hemos definido una lista de valores que tomará la función GridSearch: C = [1e-4, 1e-3, 0.01, 0.1, 0.2, 0.5].
- **Class_weight:** define el peso de cada clase. En nuestro caso, se ha asignado el valor *balanced*. Usa los valores de las etiquetas asignadas manualmente en 4.3 para ajustar estos pesos de forma inversamente proporcional a la frecuencia de cada clase en los datos de entrada.

4.6.3 K Vecinos más próximos

Se trata del método KNeighborsClassifier de la librería *Sklearn.neighbors*. Los hiper-parámetros que tenemos en cuenta en GridSearch para este clasificador son:

- **N_neighbors:** indica la cantidad de vecinos que se tendrán en cuenta para clasificar un tuit como *bullying* o no, como se explica en 3.2.6.3. Probamos desde 1 hasta 30 vecinos.
- **Algorithm:** el método usado para calcular distancias entre documentos. Se define como *auto*, que trata de decidir cuál es el mejor entre las posibilidades que oferta: *ball_tree, kd_tree* y búsqueda por fuerza bruta.
- **Weights:** es la función de pesos que se usa en la predicción. Los valores que se han probado han sido *uniform*, que trata a todos los vecinos por igual, y *distance*, que hará que los vecinos más cercanos influyan más que los más lejanos en la decisión final.

4.6.4 Random Forest

Este clasificador se inicializa en la librería con el método *RandomForestClassifier*. Los parámetros que se han considerado importantes para este se detallan a continuación:

- **N_jobs:** Se define como -1, para que los procesos que se ejecuten en paralelo sean el número de núcleos.
- **Max_features:** indica el número de atributos que tiene en cuenta cuando busca la mejor manera de dividir el conjunto de datos. Se fija al valor sqrt¸ por lo que $\text{max\_features} = \sqrt{número\ de\ atributos}$
- **N_estimators:** es el número de árboles en el bosque. Se prueban los diferentes resultados con la lista [9, 18, 27, 36, 45, 54, 63].
- **Max_depth:** es la máxima profundidad que puede alcanzar el árbol. El GridSearch investigará cuál es la mejor de entre [1, 5, 10, 15, 20, 25, 30].
- **Min_samples_leaf:** indica el número mínimo de muestras (samples) necesarias para poder considerar al tuit como un nodo hoja (ver Ilustración 16). La lista [1, 2, 4, 6, 8, 10] incluye las variantes para GridSearch.

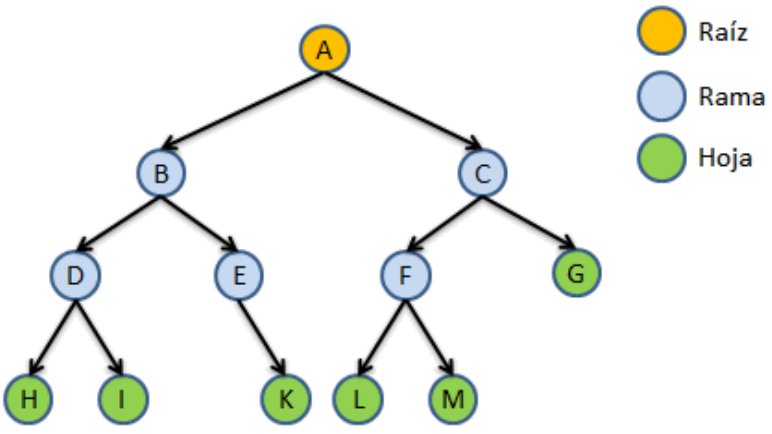

Ilustración 16. Tipos de nodos en un árbol.

- **Class_weight:** funciona igual que el parámetro de mismo nombre del clasificador explicado en 4.6.2, definiendo el peso de cada clase. También en este caso se ha declarado como balanced.

5 Pruebas y resultados

A continuación, se presentan todas las pruebas realizadas con cada clasificador. Para comparar sus rendimientos, estudiaremos las métricas de **precisión y recall**, básicas en sistemas de búsqueda y minería de información. Serán calculadas a partir de su matriz de confusión, una representación gráfica del funcionamiento de clasificadores de aprendizaje supervisado, como se trata en este trabajo. Las filas muestran los valores reales, mientras que las columnas contienen los valores de cada clase que el clasificador ha predicho, como se muestra en la Ilustración 17, donde 288 casos son correctamente clasificados como no *bullying*, 541 como *bullying*, 23 son incorrectamente clasificados como *bullying* y 65 son realmente *bullying*, aunque se prediga lo contrario.

		Valor predicho	
		No bullying	Bullying
Valor real	No bullying	288	23
	Bullying	65	541

Ilustración 17. Ejemplo de matriz de confusión.

Si vemos la matriz de confusión como se muestra en la Ilustración 18:

		Valor predicho	
		No bullying	Bullying
Valor real	No bullying	Verdadero negativo (True Negative, TN)	Falso Positivo (False Positive, FP)
	Bullying	Falso Negativo (False Negative, FN)	Verdadero positivo (True Positive, TP)

Ilustración 18. Matriz de confusión con TN, FP, FN y TP.

Entonces podemos representar la precisión y recall como:

$$Precision = \frac{TP}{TP + FP}$$

Ecuación 7. Precisión de un clasificador

$$Recall = \frac{TP}{TP + FN}$$

Ecuación 8. Recall de un clasificador.

La precisión indica la cantidad de documentos devueltos que son relevantes para la consulta realizada. En este caso, se refiere a los tuits que realmente son *bullying* de entre todos los que se han predicho como tal.
Por su parte, el recall, o exhaustividad, viene a expresar la proporción de los tuits que se han predicho como *ciberbullying* frente a todos los etiquetados realmente como ciberacoso.

Teniendo definidas estas métricas, tenemos que tener en mente que la prioridad es tener la menor cantidad posible de falsos negativos, ya que así evitamos pasar por alto algún caso de acoso virtual. En nuestro caso, un clasificador será mejor cuanto mayor sea el recall. También tendremos en cuenta la curva ROC (*Receiving Operating Characteristic*), otra forma visual de ver cómo funciona un clasificador binario. En particular, compara la tasa de verdaderos positivos (TP) y la de falsos positivos (FP). Cuanto mayor sea la AUC (*Area Under the Curve*), es decir, el área bajo la curva, mejor será el clasificador.

5.1 Naive Bayes

5.1.1 Multinomial

- **BoW**

Tras cargar el corpus de tuits etiquetados, y limpiar el texto, analizamos el GridSearch sobre MultinomialNB con los parámetros comentados en 4.6.1. El resultado proclama como mejores valores: {'alpha': 1.0, 'class_prior': [0.65, 0.35], 'fit_prior': True}. Por tanto, definimos el clasificador de esa manera. Los resultados de la matriz de confusión y la curva ROC se muestran en la Ilustración 19, donde observamos que el área bajo la curva (AUC) es de 0.68.

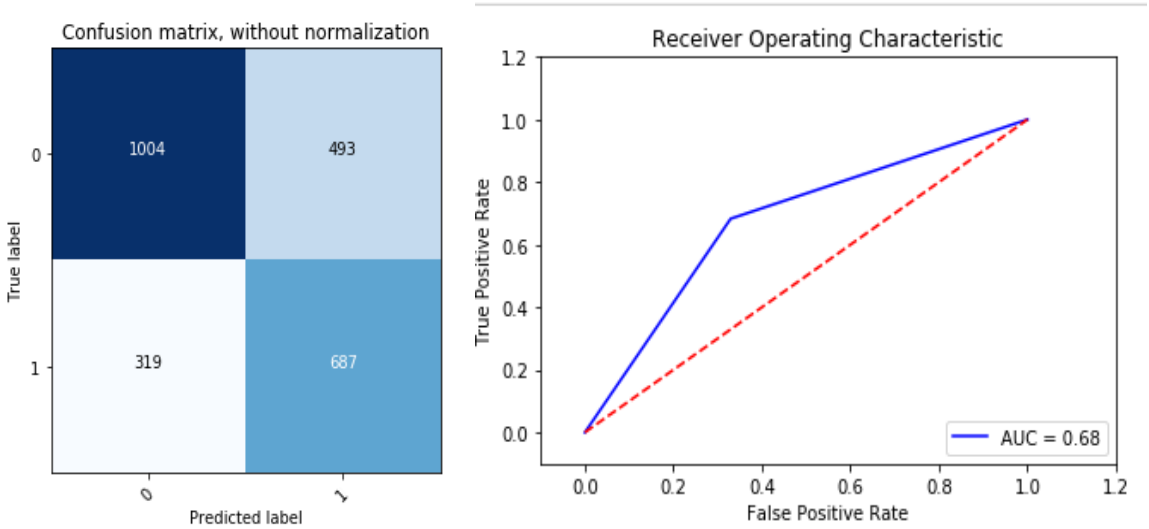

Ilustración 19. Resultados de Naive Bayes Multinomial con BoW.

Además, las métricas de precisión y recall resultan ser:

$$Prec = \frac{687}{687 + 493} = 0.5822 \qquad Recall = \frac{687}{687 + 319} = 0.6829$$

- **TF-IDF con unigramas**

El GridSearch detecta el mejor funcionamiento de este Naive Bayes Multinomial con los valores {'alpha': 1.0, 'class_prior': [0.55, 0.45], 'fit_prior': True}. Los resultados para los mejores parámetros comentados son mostrados en las dos figuras de la Ilustración 20.

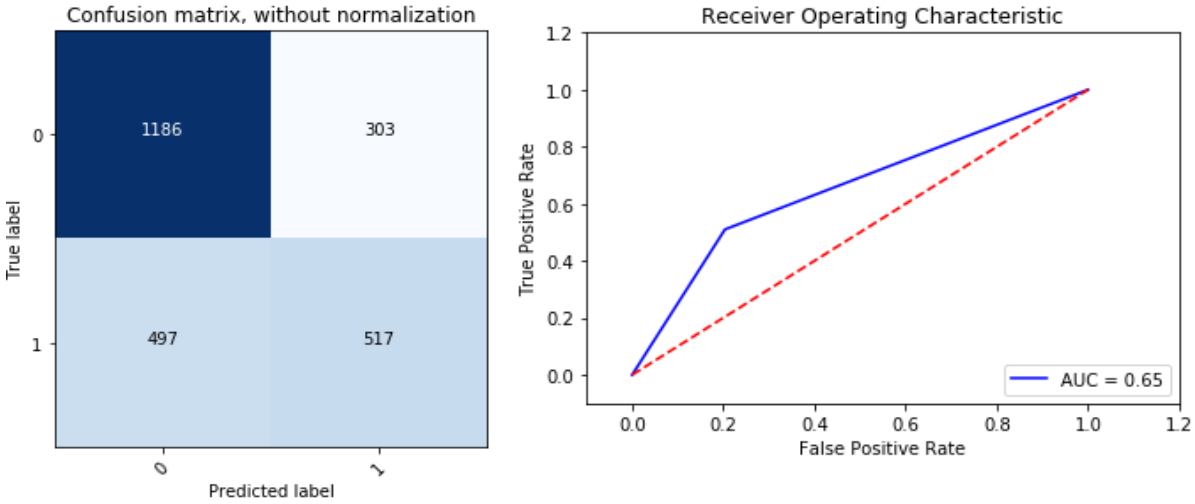

Ilustración 20. Resultados para Naive Bayes Multinomial con unigramas.

Por tanto, los cálculos de precisión y recall son:

$$Prec = \frac{517}{517 + 303} = 0.6305 \qquad Recall = \frac{517}{517 + 497} = 0.5098$$

- **TF-IDF con bigramas**

Los mejores parámetros coinciden con los mejores en TF-IDF con unigramas. Por tanto, los resultados son bastante similares, como se ve en la Ilustración 21.

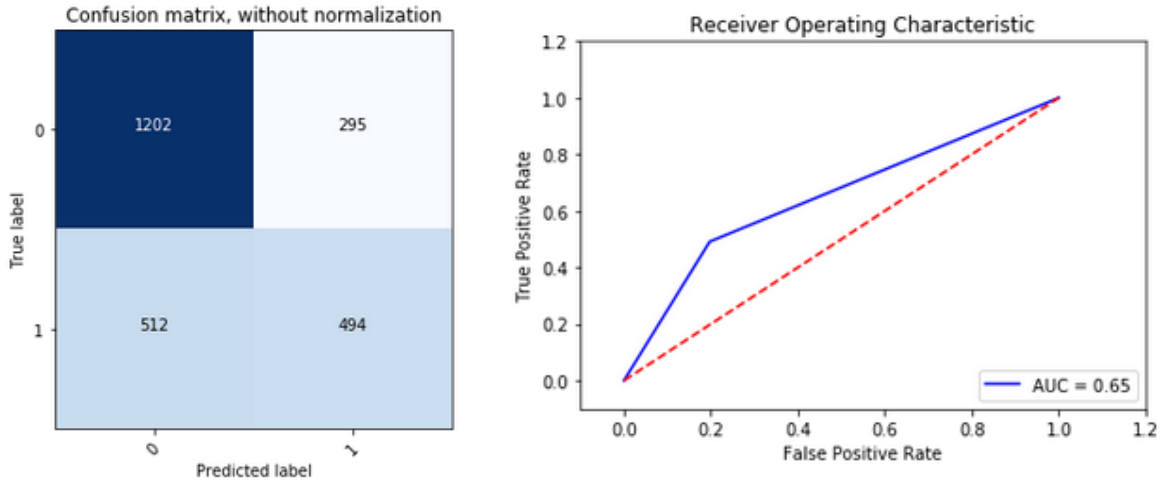

Ilustración 21. Resultados para Naive Bayes Multinomial con bigramas.

Las métricas de precisión y recall que se obtienen son:

$$Prec = \frac{494}{494 + 295} = 0.6261 \qquad Recall = \frac{494}{494 + 512} = 0.491$$

5.1.2 Bernoulli

- **BoW**

En este caso los mejores hiper-parámetros son {'alpha': 0.8, 'class_prior': [0.7, 0.3], 'fit_prior': True}. La Ilustración 22 muestra los resultados para el clasificador de Naive Bayes tras convertir los textos a vectores mediante BoW.

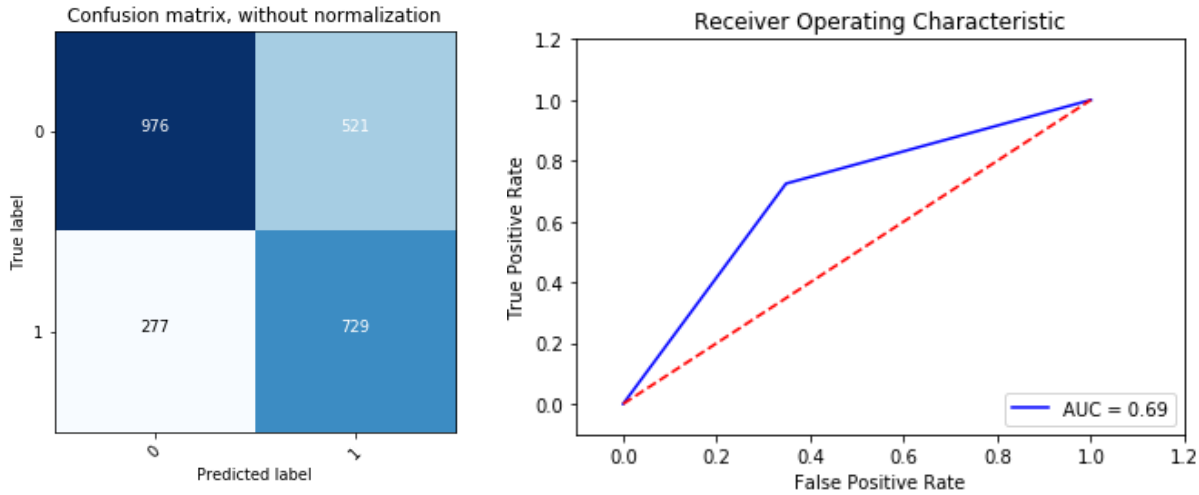

Ilustración 22. Resultados de Naive Bayes Bernoulli con BoW.

Las métricas conseguidas son:

$$Prec = \frac{729}{729 + 521} = 0.5832 \qquad Recall = \frac{729}{729 + 277} = 0.72465$$

- **TF-IDF con unigramas**

GridSearch sobre BernoulliNB tras convertir los tuits a vectores mediante TF-IDF con unigramas decide que los hiper-parámetros adecuados son {'alpha': 0.7, 'class_prior': [0.7, 0.3], 'fit_prior': True}. La Ilustración 23 presenta los resultados para estos valores.

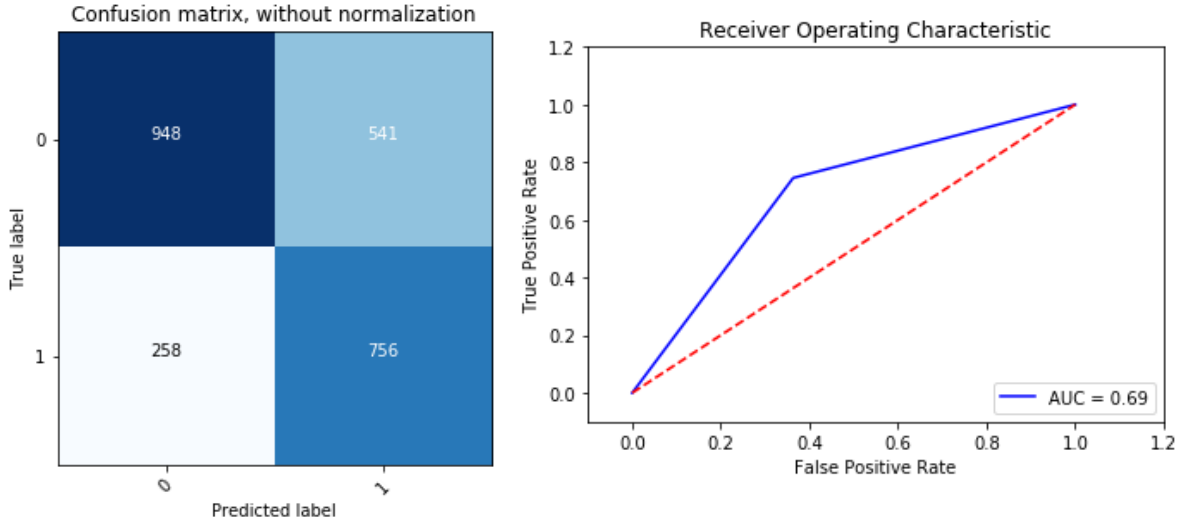

Ilustración 23. Resultados para Naive Bayes Bernoulli con unigramas.

La precisión y recall en este caso son:

$$Prec = \frac{756}{756 + 541} = 0.5828 \qquad Recall = \frac{756}{756 + 258} = 0.7455$$

- **TF-IDF con bigramas**

El cambio respecto a unigramas devuelto por el GridSearch es que es preferible Alpha=0.5. Se exponen los rendimientos en la Ilustración 24, donde podemos ver que son inferiores a los obtenidos con unigramas, así como su precisión y recall.

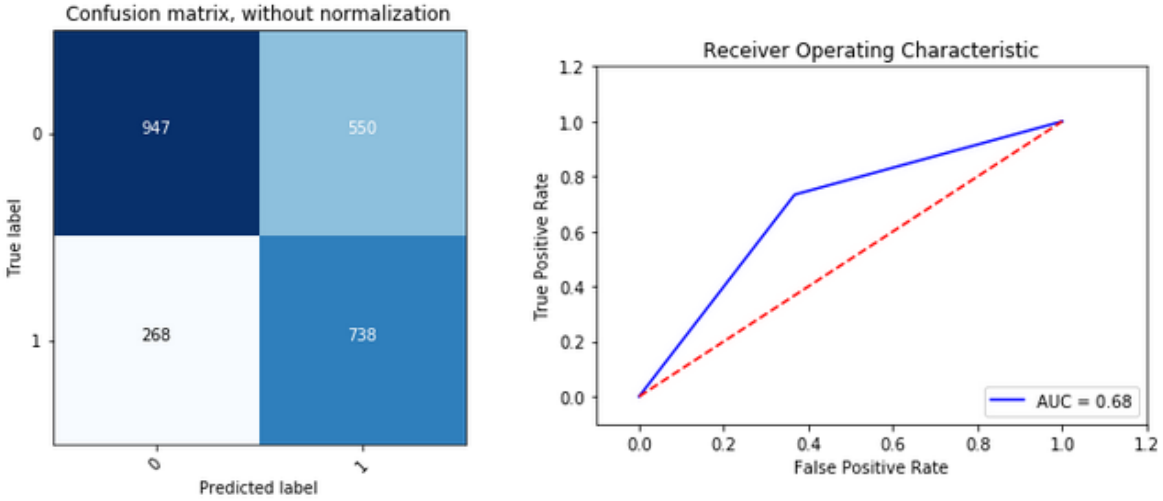

Ilustración 24. Resultados para Naive Bayes Bernoulli con bigramas.

$$Prec = \frac{738}{738 + 550} = 0.5729 \qquad Recall = \frac{738}{738 + 268} = 0.7335$$

5.2 Random Forest

- **BoW**

Con los parámetros con valores {'max_depth': 30, 'min_samples_leaf': 1, 'n_estimators': 9} y class_weight='balanced'. Se revelan las conclusiones de haberlos aplicado en la Ilustración 25.

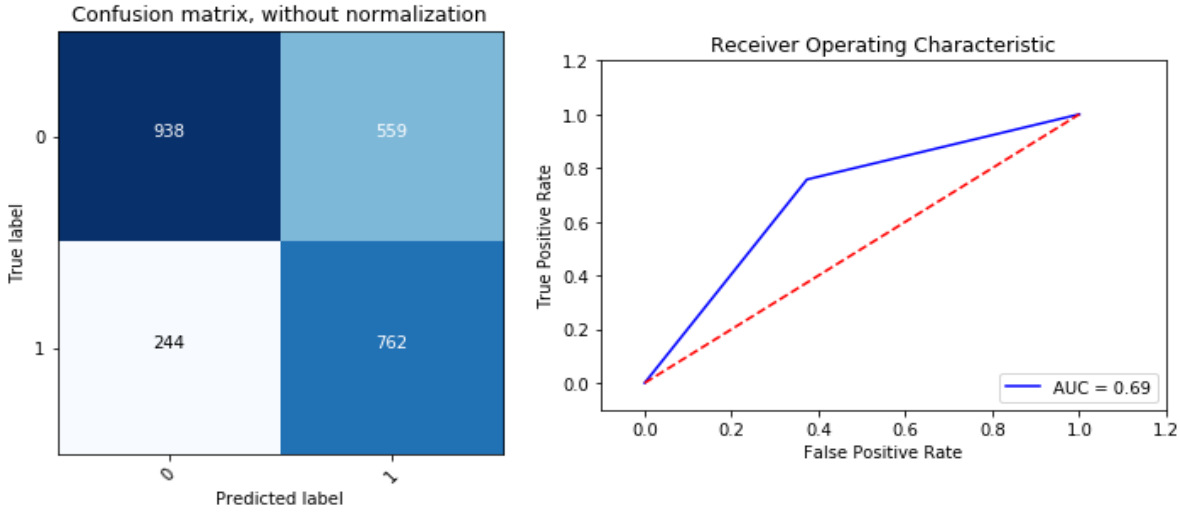

Ilustración 25. Resultados de Random Forest con BoW.

Las métricas derivadas de la matriz de confusión son:

$$Prec = \frac{762}{762 + 559} = 0.5768 \qquad Recall = \frac{762}{762 + 244} = 0.7574$$

- **TF-IDF con unigramas**

Los valores seleccionados son {'max_depth': 30, 'min_samples_leaf': 1, 'n_estimators': 36}. Tras aplicarlos, exhibimos las conclusiones mediante las figuras de la Ilustración 26.

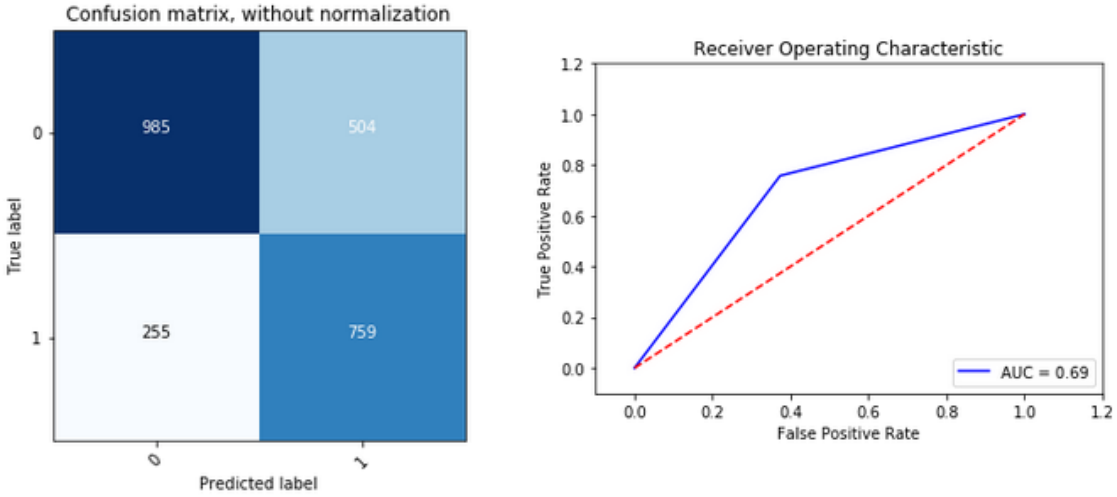

Ilustración 26. Resultados para Random Forest con unigramas.

Además, la preicisión y el recall son:

$$Prec = \frac{759}{759 + 504} = 0.6009 \qquad Recall = \frac{759}{759 + 255} = 0.7485$$

- **TF-IDF con bigramas**

En este caso, {'max_depth': 30, 'min_samples_leaf': 2, 'n_estimators': 36} es el conjunto ganador para el GridSearch. Así, la matriz de confusión muestra un bajo número de falsos negativos, y el área bajo la curva (AUC) es de 0.69. Lo podemos observar en la Ilustración 27.

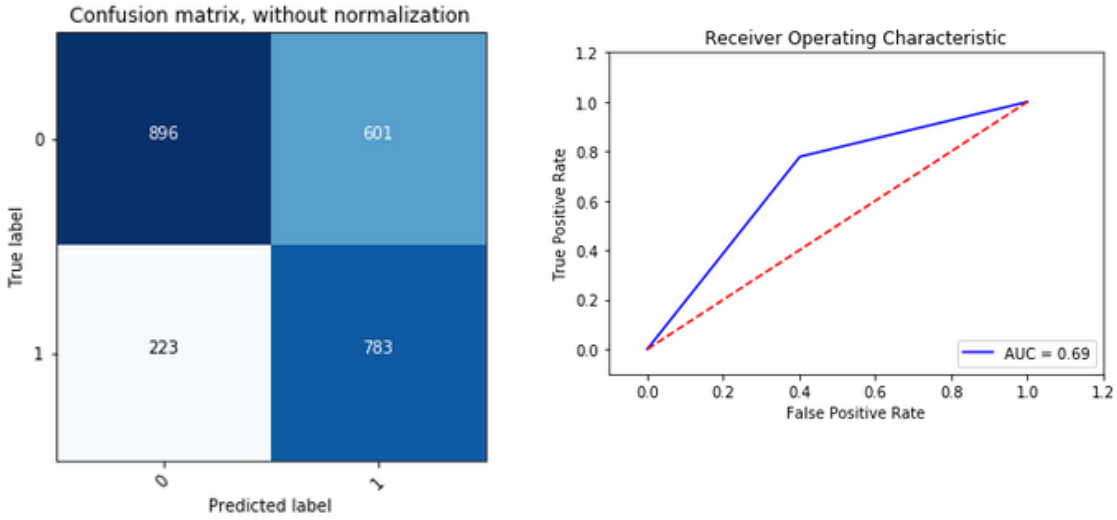

Ilustración 27. Resultados para Random Forest con bigramas.

Conseguimos, por tanto, con Random Forest y bigramas para TF-IDF los siguientes valores en cuanto a precisión y exhaustividad:

$$Prec = \frac{783}{783 + 601} = 0.5657 \qquad Recall = \frac{783}{783 + 223} = 0.7783$$

5.3 Support Vector Machines

- **BoW**

Probamos con los parámetros comentados en la fase de desarrollo, consiguiendo el mejor resultado para {'C': 1, 'class_weight': 'balanced', 'gamma': 0.1, 'kernel': 'rbf'}, que se muestra en la Ilustración 28.

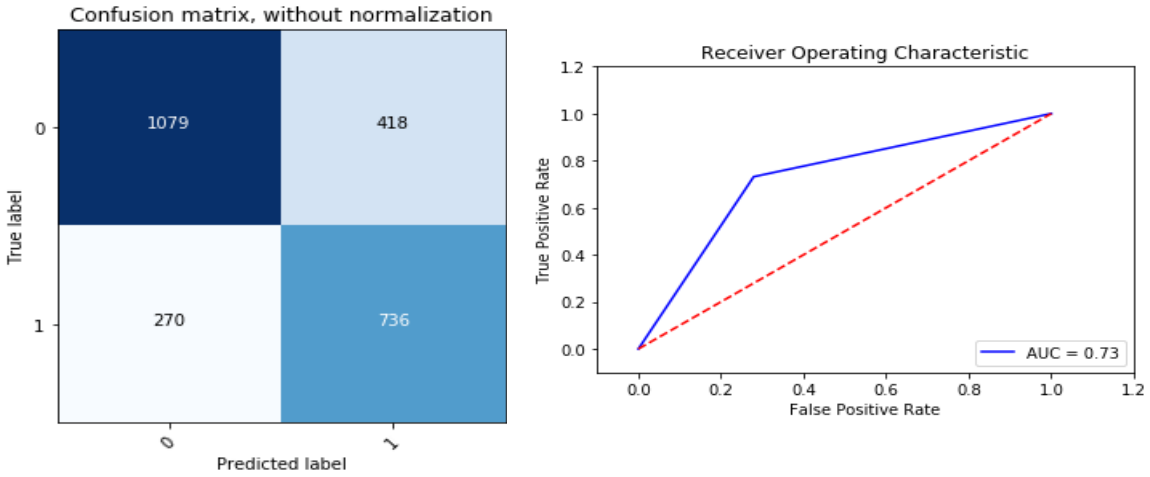

Ilustración 28. Resultados de SVM con BoW.

$$Prec = \frac{736}{736 + 418} = 0.6377 \qquad Recall = \frac{736}{736 + 270} = 0.7316$$

- **TF-IDF con unigramas**

Los resultados que se muestran se dan con los valores en los parámetros de entrada de este clasificador {'C': 200.0, 'class_weight': 'balanced', 'gamma': 0.001, 'kernel': 'rbf'}.

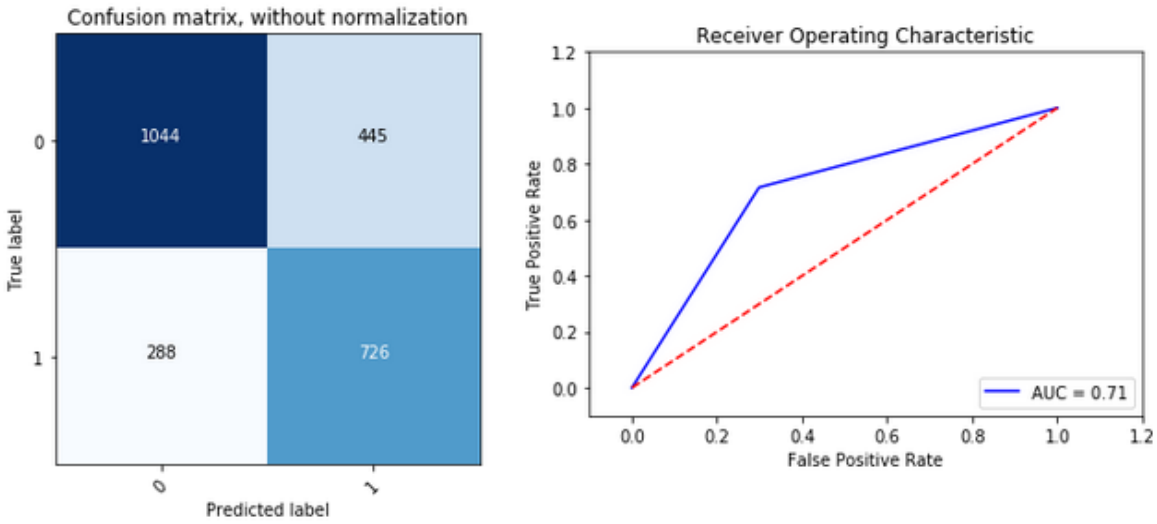

Ilustración 29. Resultados para SVM con unigramas.

La Ilustración 29 presenta los resultados para este clasificador, esta forma de conversión a vectores, junto con la aplicación de los mejores hiperparámetros.
Las métricas resultan ser:

$$Prec = \frac{726}{726 + 445} = 0.6199 \qquad Recall = \frac{726}{726 + 288} = 0.7159$$

- **TF-IDF con bigramas**

Al acabar la búsqueda del GridSearch, decide que, en este caso, los mejores parámetros se corresponden con: {'C': 2000.0, 'class_weight': 'balanced', 'gamma': 0.0001, 'kernel': 'rbf'}. La matriz de confusión que se obtiene, junto con la curva ROC se muestran en la Ilustración 30.

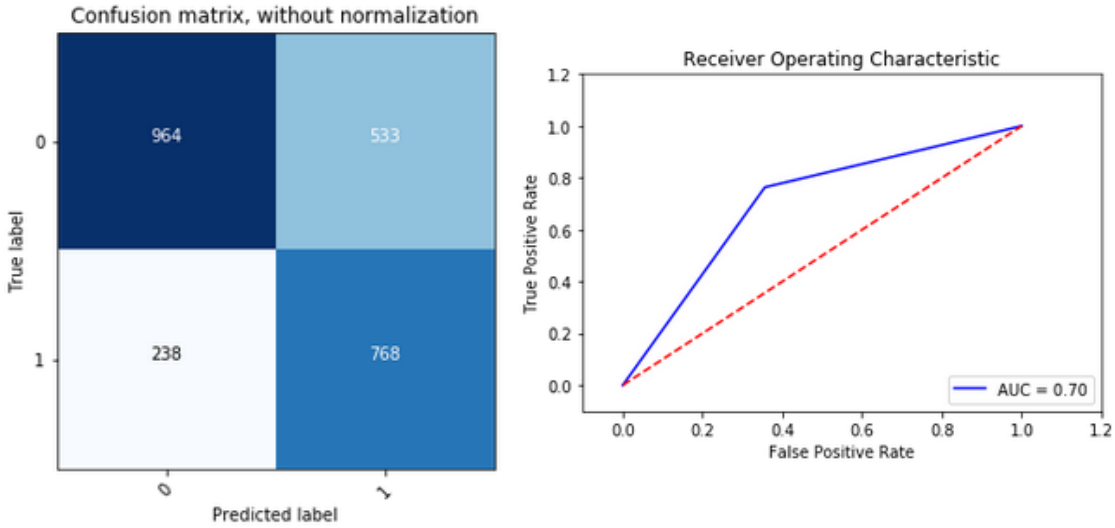

Ilustración 30. Resultados para SVM con bigramas.

La precisión y recall conseguidas son:

$$Prec = \frac{768}{768 + 533} = 0.5903 \qquad Recall = \frac{768}{768 + 238} = 0.7634$$

5.4 K Vecinos más próximos

- **BoW**

La combinación de parámetros {'algorithm': 'auto', 'n_neighbors': 4, 'weights': 'uniform'} es la que resulta ganadora para este clasificador. La matriz de confusión y la ROC están presentes en la Ilustración 31.

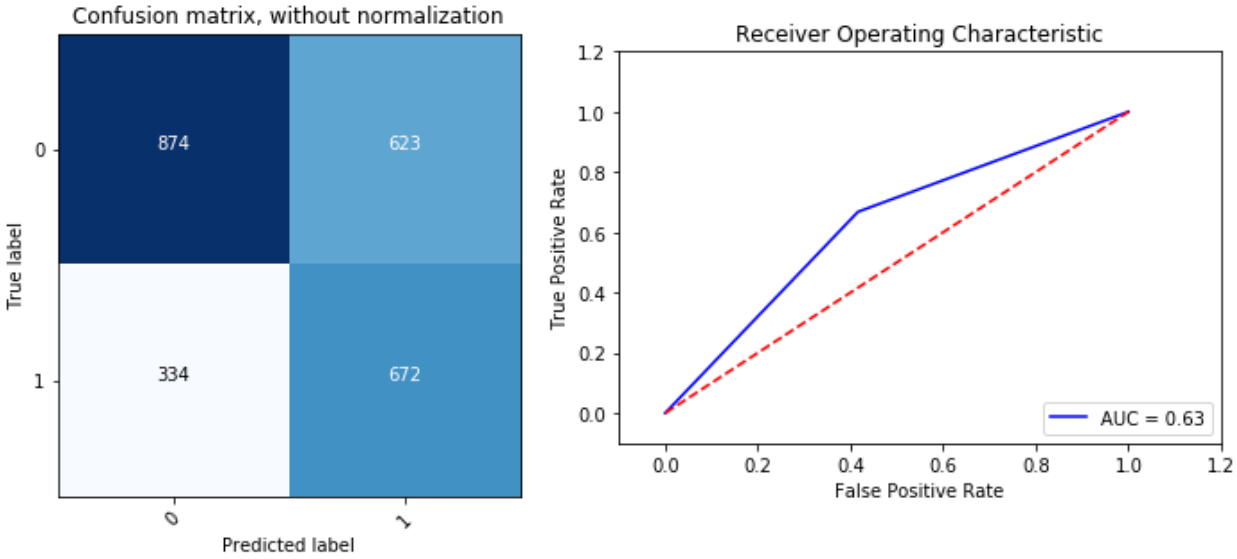

Ilustración 31. Resultados para kNN con BoW.

Los resultados para las métricas son, por tanto:

$$Prec = \frac{672}{672 + 623} = 0.5189 \qquad Recall = \frac{672}{672 + 334} = 0.6679$$

- **TF-IDF con unigramas**

El método GridSearch decide que 28 vecinos para tener en cuenta y weights= 'uniform', explicado en 4.6.3, devuelve el más alto de los rendimientos para este clasificador.

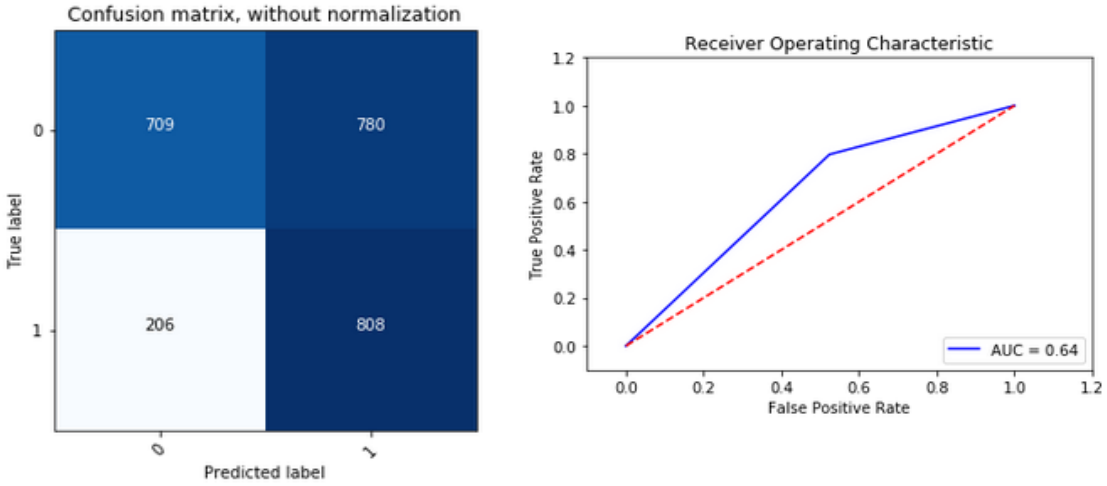

Ilustración 32. Resultados para kNN con unigramas.

La Ilustración 32 contiene la matriz de confusión y la curva ROC correspondiente. De ella, obtenemos la precisión y el recall:

$$Prec = \frac{808}{808 + 780} = 0.5088 \qquad Recall = \frac{808}{808 + 206} = 0.7968$$

- **TF-IDF con bigramas**

La combinación de parámetros {'algorithm': 'auto', 'n_neighbors': 24, 'weights': 'uniform'} es la devuelta por GridSearch como óptima para este clasificador. La Ilustración 33 presenta las figuras correspondientes.

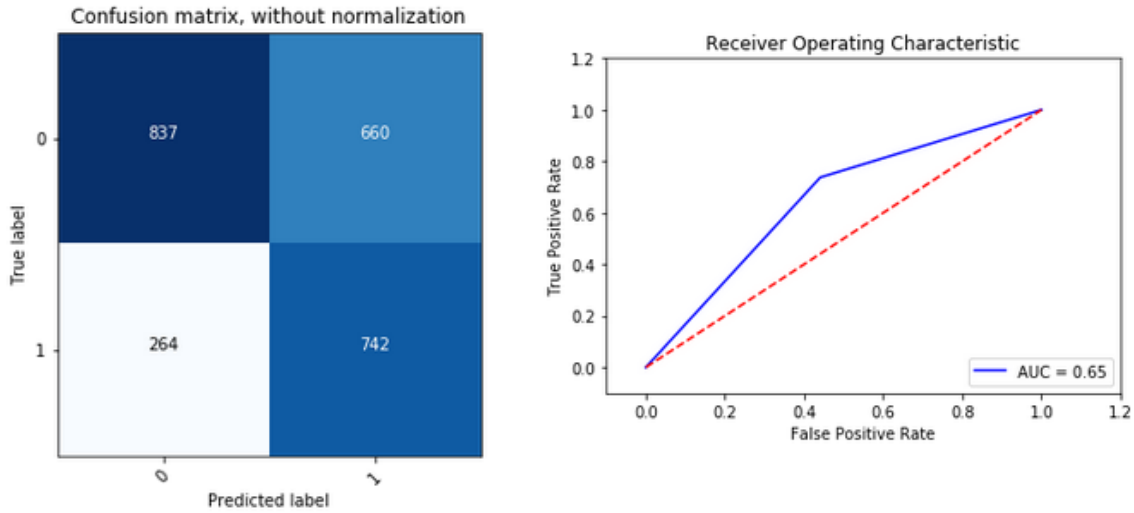

Ilustración 33. Resultados para kNN con bigramas.

Por último, las métricas de precisión y exhaustividad resultantes son:

$$Prec = \frac{742}{742 + 660} = 0.5292 \qquad Recall = \frac{742}{742 + 264} = 0.7375$$

5.5 Análisis de resultados

Tras toda la fase de pruebas con los diferentes clasificadores, es momento de comparar todos los resultados. En las tablas 4, 5 y 6 se presentan de manera esquematizada tanto la precisión como el recall de todas las posibles variaciones expuestas en Pruebas y resultados; así como el área bajo la curva ROC (AUC).

	BoW	unigramas	bigramas
NB Multinomial	0,5822	0,6305	0,6261
NB Bernoulli	0,5832	0,5828	0,5729
Random Forest	0,5768	0,6009	0,5657
SVM	0,6377	0,6199	0,5903
kNN	0,5189	0,5088	0,5292

Tabla 4. Comparativa de precisión.

En estas tres tablas, las filas representan los distintos algoritmos, mientras que las columnas se corresponden con la información sobre los métodos de conversión a vectores numéricos: bolsa de palabras, TF-IDF con unigramas y con bigramas, respectivamente.

	BoW	unigramas	bigramas
NB Multinomial	0,6829	0,5098	0,491
NB Bernoulli	0,72465	0,7455	0,7335
Random Forest	0,7574	0,7485	0,7783
SVM	0,7316	0,7159	0,7634
kNN	0,6679	0,7968	0,7375

Tabla 5. Comparativa de recall.

Por ejemplo, en la Tabla 6, se puede observar que para Random Forest se obtienen los mismos resultados para el área bajo la curva independientemente del método de conversión a vectores empleado.

	BoW	unigramas	bigramas
NB Multinomial	0,68	0,65	0,65
NB Bernoulli	0,69	0,69	0,68
Random Forest	0,69	0,69	0,69
SVM	0,73	0,71	0,7
kNN	0,63	0,64	0,65

Tabla 6. Comparativa de área bajo la curva ROC (AUC).

Dado que el objetivo principal es detectar *bullying*, es preferible tener un número bajo de falsos negativos, ya que no queremos pasar por alto la detección de los tuits en los que está presente. Este hecho afecta directamente al recall: se buscará el más alto. Aun así, es recomendable que se combine con una precisión razonable. Es por eso por lo que los clasificadores con mejores resultados son: Random Forest con TF-IDF mediante bigramas, SVM con BoW, y SVM con unigramas, ya que proporcionan, por una parte, un recall de 0.7783, 0.7316 y 0.7159, respectivamente, y, por otra parte, una precisión de 0.5657, 0.6377 y 0.6199.

Para decidir cuál podría ser el mejor de entre los tres elegidos, observamos sus matrices de confusión del capítulo anterior, fijándonos en la celda de falsos negativos (FN). Además, es importante el valor de AUC. Cuanto más cercano sea a 1, mayor indicador de buen clasificador será. [29]
En definitiva, podemos considerar a **SVM con BoW** el mejor clasificador de entre todas las pruebas realizadas en este trabajo. Su precisión = 0.6377, su recall = 0.7316, y su AUC = 0.73.

Estudiemos a fondo los resultados obtenidos con este clasificador. El método GridSearch nos devolvió como mejores parámetros los mostrados en la Tabla 7.

C	class_weight	gamma	kernel
1	balanced	0,1	RBF

Tabla 7. Parámetros elegidos por el GridSearch para SVM con BoW.

La desviación estándar de esta elección se sitúa en 0.01. La Tabla 8 expone un informe más detallado de clasificación, para ambas clases (0 = no *bullying*, 1 = *bullying*) e incluyendo F1-Score. Esta métrica, denominada también media armónica, es utilizada para expresar el rendimiento de un clasificador. Es una manera de relacionar precisión y recall, y se calcula como:

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

Ecuación 9. F1 o media armónica.

Esta medida alcanzará el mejor valor en 1, con perfecta precisión y recall. Los resultados para ambas clases se muestran en la Tabla 8. En ella también se muestra que este clasificador final cuenta con una precisión del 0.8 para la clase de *no bullying*, y un recall del 0.72. Para la clase que detecta *bullying* se muestra una precisión del 0.64 y un recall del 0.73.

	Precision	Recall	F1-Score	Support
0	0,8	0,72	0,76	1497
1	0,64	0,73	0,68	1008
media / total	0,73	0,73	0,73	2503

Tabla 8. Informe de clasificación para SVM con BoW.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este trabajo se ha abordado la detección del ciberacoso mediante el tratamiento de mensajes obtenidos de la red social Twitter con técnicas del procesamiento de lenguaje natural y la aplicación de algoritmos de aprendizaje automático para clasificar dichos mensajes entre mensajes que reflejan acoso y mensajes que no lo reflejan.

Como se ha explicado en detalle en el análisis de resultados (apartado 5.5), una vez tratados los mensajes utilizando la lematización, eliminación de *stopwords*, detección de expresiones regulares y tras la aplicación de distintos algoritmos de clasificación (Naive Bayes, SVM, Random Forest y kNN), se puede concluir que los mejores resultados los ofrece el algoritmo Support Vector Machines con BoW. Se consigue un grado de precisión de casi 64% y un recall (exhaustividad) del 73%.

Estos resultados se reflejan en que de entre los 1.006 mensajes etiquetados como *bullying* dentro del conjunto de prueba, este clasificador predice que 270 no lo son (ver Ilustración 28). A pesar de conseguir las mejores métricas respecto al resto de opciones, son números que se pueden mejorar. Abordaremos las posibles mejoras y el trabajo futuro en el apartado 416.2.

Este trabajo me ha permitido adentrarme en el ámbito del aprendizaje automático y del procesamiento del lenguaje natural, y aprender distintas técnicas y algoritmos desconocidos para mí hasta el momento. También he puesto en práctica conocimientos relacionados con la búsqueda y minería de datos, explorando la API de Twitter y obteniendo información en esos mensajes que se ha combinado con los clasificadores estudiados. Además, he podido aplicar una metodología de trabajo: analizando primero el problema a resolver y los requisitos a satisfacer, investigando las distintas posibilidades, hasta obtener los resultados y conclusiones presentadas.

En el futuro se podría investigar la posibilidad de mejorar los resultados obtenidos explorando algunas alternativas y variaciones sobre el trabajo realizado como las que se describen en el siguiente apartado.

6.2 Trabajo futuro

En prácticamente todas las etapas de desarrollo del trabajo hay variantes que se pueden abordar si se sigue trabajando en este tema con el fin de mejorar el rendimiento conseguido.

El proceso de obtención de textos se pueden explorar otras redes sociales de las que se puedan conseguir los mensajes de los usuarios. Con respecto a la obtención de mensajes de Twitter, también se podría considerar la posibilidad de incluir más palabras clave para buscar en la plataforma mediante la API. Ampliando el conjunto total de textos, conseguimos que los resultados del clasificador funcionen de forma general para más situaciones de ciberacoso.

Además, sería una buena idea estudiar los textos que se van consiguiendo, analizando las palabras más comunes en mensajes agresivos de acoso e incluirlas en el proceso de filtrado. Con ello se podrían conseguir deducciones más concretas.

Como ya se comentó en el capítulo 3.2.4, añadir más expertos etiquetadores al proceso sería de gran ayuda. Por una parte, se repartiría la carga de trabajo que implica el etiquetado de mensajes. Por otra parte, se reducirían la posible subjetividad y los efectos derivados de apoyar todas las decisiones sobre una sola persona.

En el tratamiento de los textos, se podría considerar otro lematizador distinto al elegido en este trabajo. Otra posibilidad es tener en cuenta los retuits (RT), para darle más relevancia a aquellos mensajes que son apoyados por más usuarios, y que podrían acarrear peores consecuencias a las víctimas. Asimismo, y mediante los campos que devuelve la API de Twitter, se podría hacer un seguimiento de las conversaciones (mediante las respuestas y el momento de publicación del tuit) para tomar como un único texto toda la conversación, y ampliar la longitud de los documentos.

Referencias

[1] J. Pérez Porto, M. Merino. Definición de acoso. Disponible en: https://definicion.de/acoso/ Publicado en 2012, actualizado en 2014.

[2] E. Solberg, D. Olweus, Prevalence Estimation of School Bullying With the Olweus Bully/Victim Questionnaire Aggressive Behavior 29(3), 2003, 239-268

[3] Fundación ANAR, I Estudio sobre el Bullying, Abril 2016.

[4] D. Olweus, Cyberbullying, an overrated phenomenon?,European Journal of Developmental Psychology 9(5), Septiembre 2012 520-538

[5] Red Social Twitter: https://twitter.com

[6] Kowalski, Giumetti, Schroeder, Lattanner, Bullying in the Digital Age: A Critical Review and Meta-Analysis of Cyberbullying Research Among Youth, Psychological Bulletin, Vol 140(4). 2014, 1073-1137

[7] Fundación ANAR, II Informe estudio sobre el Ciberbullying, Septiembre 2016.

[8] Límites de búsqueda en Twitter. Disponible en: https://developer.twitter.com/en/docs/basics/rate-limiting . Accedido en Mayo, 2018.

[9] J. W. Patchin, S. Hinduja, Cyberbullying Prevention and Response: Expert Perspectives. Routledge, 2012.

[10] D. Boyd, Taken Out of Context: American Teen Sociality in Networked Publics. DOI: http://dx.doi.org/10.2139/ssrn.1344756. 2008.

[11] E. Saneleuteiro, R. López-García-Torres, Violencia escolar: derechos y deberes para la convivencia. Tendencias Pedagógicas. http://dx.doi.org/10.15366/tp2017.30.015. 2017.

[12] J.A. Luengo Latore, Hacia una ética de las relaciones en las redes sociales: Guía de recursos didácticos para Centros Educativos. Colegio Oficial de Psicólogos de Madrid. ISBN (13): 978-84-87556-60-9. 2014.

[13] M. Gámez-Guadix, I. Orue., Smith, P. K. y Calvete, E. Longitudinal and Reciprocal Relations of Cyberbullying with Depression, Substance Use, and Problematic Internet Use among Adolescents. Journal of Adolescent Health, 53, 2013. 446 -452.

[14] B. Ballesteros, Jóvenes: bullying y ciberbullying. Teléfono ANAR: la herramienta de prevención, detección e intervención frente al acoso escolar y el ciberbullying en España, 2017.

[15] S. Limber. Cyber Bullying: A Prevention Curriculum for Grades 3-5 y Cyber Bullying: A Prevention Curriculum for Grades 6-12. Hazelden Publishing. 2016.

[16] Datos sobre el *bullying* y fuentes para prevenirlo. Disponible en: https://educateforaction.com/bullying/ . Accedido en Mayo, 2018.

[17] Lista AFINN de palabras con puntuaciones. Se encuentra en: http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010. 2015.

[18] K. Reynolds, A. Kontostathis, Using Machine Learning to Detect Cyberbullying, Proceedings of the 10th International Conference on Machine Learning and Applications and Workshops, Vol 02. 2011, 241-244

[19] L. Engman, Automatic Detection of Cyberbullying on Social Media. Tesis de master. Universidad de Umea, 2016.

[20] The Ditch Label, The Annual Bullying Survey 2017. Disponible en: https://www.ditchthelabel.org/wp-content/uploads/2017/07/The-Annual-Bullying-Survey-2017-1.pdf. Accedido en Mayo, 2018.

[21] S. Kim, K. Han, Some Effective Techniques for Naive Bayes Text Classification. IEEE Transactions on Knowledge and Data Engineering, 18(11). 2006. 1457 - 1466

[22] Funcionamiento de Naive Bayes, se encuentra en la web: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ . Accedido en Mayo, 2018.
[23] Teorema de Bayes, disponible en: https://es.wikipedia.org/wiki/Teorema_de_Bayes
[24] C. D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, ISBN: 978 0 521 86571 5. 2008.
[25] Funcionamiento del algoritmo doc2vec, disponible en: https://medium.com/scaleabout/a-gentle-introduction-to-doc2vec-db3e8c0cce5e
[26] Lematizador TreeTagger: http://treetaggerwrapper.readthedocs.io/en/latest/
[27] C. Hsu, C. Chang, C. Lin A Practical Guide to Support Vector Classification, Mayo 2016.
[28] M. Swant (Adweek Digital), Twitter Is Cracking Down on Trolls and Offensive Tweets With These New Tools. Febrero 2017. Disponible en: https://www.adweek.com/digital/twitter-is-cracking-down-on-trolls-and-offensive-tweets-with-these-new-tools/ Accedido en Mayo 2018.
[29] J. Llopis Pérez, Análisis ROC. Disponible en: https://estadisticaorquestainstrumento.wordpress.com/2013/02/13/tema-23-analisis-roc/ Accedido en Mayo 2018.
[30] P. Black, M. Wollis, M. Woodworth, J. Hancock, A linguistic analysis of grooming strategies of online child sex offenders: Implications for our understanding of predatory sexual behavior in an increasingly computer-mediated world. Child Abuse and Neglect, 44. 2015. 140-149
[31] R. Kowalski, G. Giumetti, A. Schroeder, M. Lattanner, Bullying in the Digital Age: A Critical Review and Meta-Analysis of Cyberbullying Research Among Youth. Psychological bulletin. 140. 10.1037/a0035618. 2014.

Glosario

API	Application Programming Interface
BoW	Bag of Words (Bolsa de Palabras)
CSV	Comma-separated values (Delimitados por comas)
Grid Search	Búsqueda exhaustiva
kNN	K Nearest Neighbors (k Vecinos más próximos)
NB	Naïve Bayes
NLP	Natural Language Processing
RT	Retuit
SVM	Support Vector Machines

Anexos

A Manual de usuario

Con el fin de hacer entender al usuario el funcionamiento del clasificador, se desarrolla este anexo.

Se adjunta el archivo **Clasificador.py**, que precisa de dos argumentos de entrada para su correcto funcionamiento. Se trata del **archivo csv** donde se encuentran los **textos ya clasificados** por el etiquetador, y el nuevo tuit, o mensaje, si se obtiene de otra red social, que se quiere clasificar.

Tras las correspondientes pruebas con los clasificadores y sus parámetros, este archivo de Python encapsula un clasificador SVM donde los textos se transforman a vectores siguiendo el método de Bag of Words. Además, ya que el primer argumento de entrada se corresponde con el total de tuits del cual se obtendrá tanto el conjunto de entrenamiento como el de prueba, en cada ejecución se procede a la limpieza de los documentos.

El segundo argumento de entrada, es decir, el nuevo tuit, se limpia y trata con el lematizador antes de introducirlo al clasificador para que devuelva la respuesta.

Este fichero devuelve, por tanto, una frase indicando si el mensaje introducido se considera bullying (1) o no (0). Se muestran un par de ejemplos a continuación:

```
C:\Users\Alejandra\Desktop\TFG_Final>python Clasificador.py textos_clasificados.
csv "JajajJAJjaa que dices subnormal, estas loco o que"
El tuit es clasificado como 1

C:\Users\Alejandra\Desktop\TFG_Final>python Clasificador.py textos_clasificados.
csv "No estoy de acuerdo con las decisiones de Pablo Iglesias.. "
El tuit es clasificado como 0
```

En el directorio donde se ha ejecutado se deben tener ambos archivos. Por último, cabe destacar que es necesario tener el lematizador TreeTagger (incluido en la carpeta TreeTagger_W) en el directorio C:/TreeTagger.

B Descarga tuits y pruebas de los clasificadores

Hasta llegar a la conclusión de que SVM con BoW es el mejor clasificador de todos los que hemos probado, ha sido necesario un conjunto de pruebas. Los "cuadernillos" de Jupyter Notebook con todas las ejecuciones, búsquedas exhaustivas y métricas concluyentes se incluyen en la carpeta "Pruebas_clasificadores".

El primer paso consiste en la obtención de los tuits de la API de Twitter. Este proceso se encuentra dentro del fichero **bajarKeyword.py**.

Por cada palabra clave de búsqueda se obtiene un archivo csv. Todos ellos están en la carpeta "Csv_descargados". Por tanto, para después pasar al etiquetado manual, ha sido necesario un proceso previo para el que se aporta el cuaderno "**Volcar_CSV**". En él, se combinan todos los archivos csv en uno solo. Además, cada documento es lematizado y filtrado con la lista de palabras que se encuentra en vocab/palabrasFiltro.txt.

El resultado de estos pasos es "output.csv", sobre el que el etiquetador podrá trabajar, incluyendo las etiquetas en la segunda columna del Excel. Tras todas estas fases, se llega a un archivo del estilo de **textos_clasificados.csv**, que será tomado como entrada por el clasificador.